Beginner’s Guide to AWS CloudTrail for Security - Ebook

Preface

Just wanted to drop a quick note and say thank you for supporting Cybr by having purchased this Ebook! I hope you enjoy it as much as I enjoyed creating it, and please don't hesitate to reach out (christophe@cybr.com) if I can help with anything at all.

Happy learning,
Christophe Limpalair
Cybr, Inc.

Copyright and notice

Version 1.0 - 11/30/2023

Copyright © 2023 by Cybr, Inc. All Rights Reserved. No part of this document may be reproduced or transmitted in any form or by any means without the prior written permission of the publisher, except in the case of brief quotations embodied in critical reviews and certain other noncommercial uses permitted by copyright law.

Every effort has been made in the preparation of this document to ensure the accuracy of the information presented. However, the information contained in this document is sold without warranty, either express or implied. Neither the authors, nor Cybr, Inc., will be held liable for any damages caused or alleged to have been caused directly or indirectly by this document.

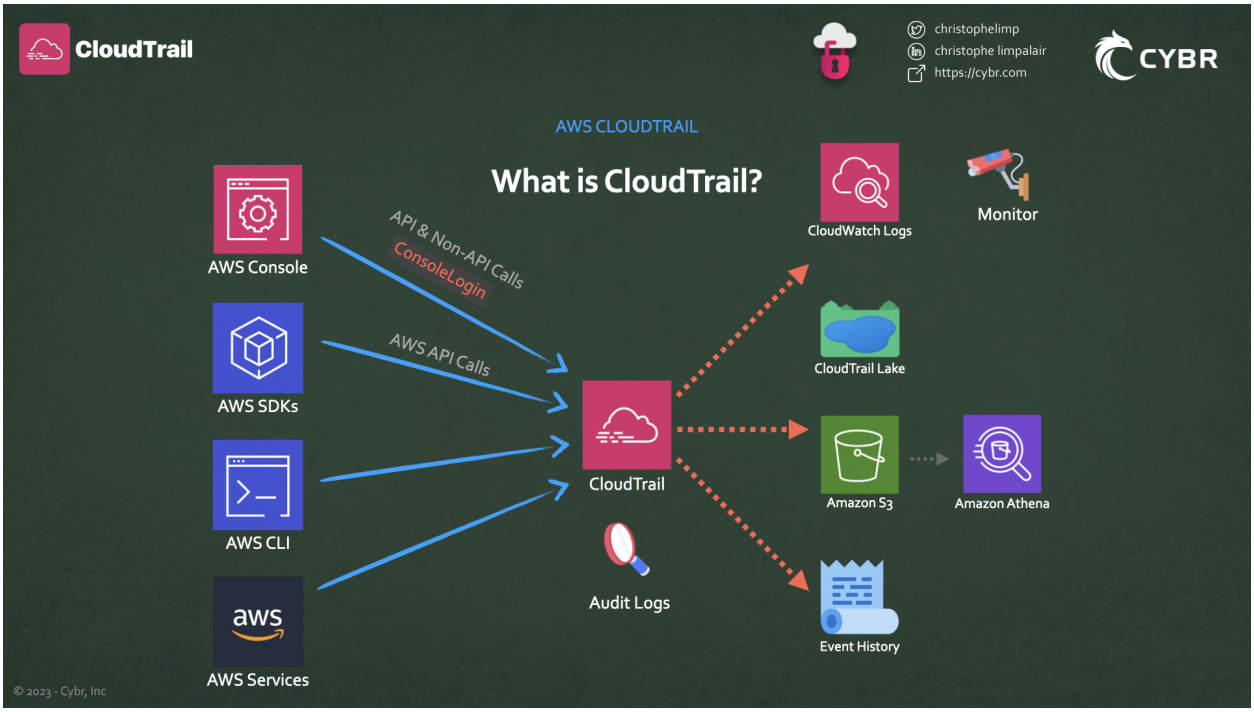
Table of Contents

Introduction

About the course and instructor

For someone getting started on their AWS security journey, if there’s one service I’d recommend that they learn very well, it would CloudTrail.

AWS CloudTrail is such an important service because it gives you visibility into your AWS account activity, and without visibility, you don’t have security.



This is a key aspect of security and operational best practices in the cloud (frankly also outside the cloud, but that's not what we're focusing on here).

So if you're serious about learning AWS security, you're in the right place, and let's take a closer look at what this course will teach you!

About your author

Before we do that, let me share just a little bit about me and who I am, and why you should take this course from me.

Hi, I'm Christophe Limpalair, and I'm the founder and an author at Cybr, where I've published many courses on topics of cloud security and ethical hacking. You may also know me from Linux Academy, where I taught multiple AWS courses including AWS certification courses. I taught for the AWS Certified Developer, AWS Certified SysOps, AWS Certified DevOps Engineer Professional, and the AWS Cloud Practitioner certifications.

There, I helped build, manage, and secure production AWS infrastructure that ran as a $1m/year budget.

Through that and other experiences, I learned how to properly design and secure AWS environments and resources. I continue to build on AWS for our platform at Cybr, and for various consulting gigs.

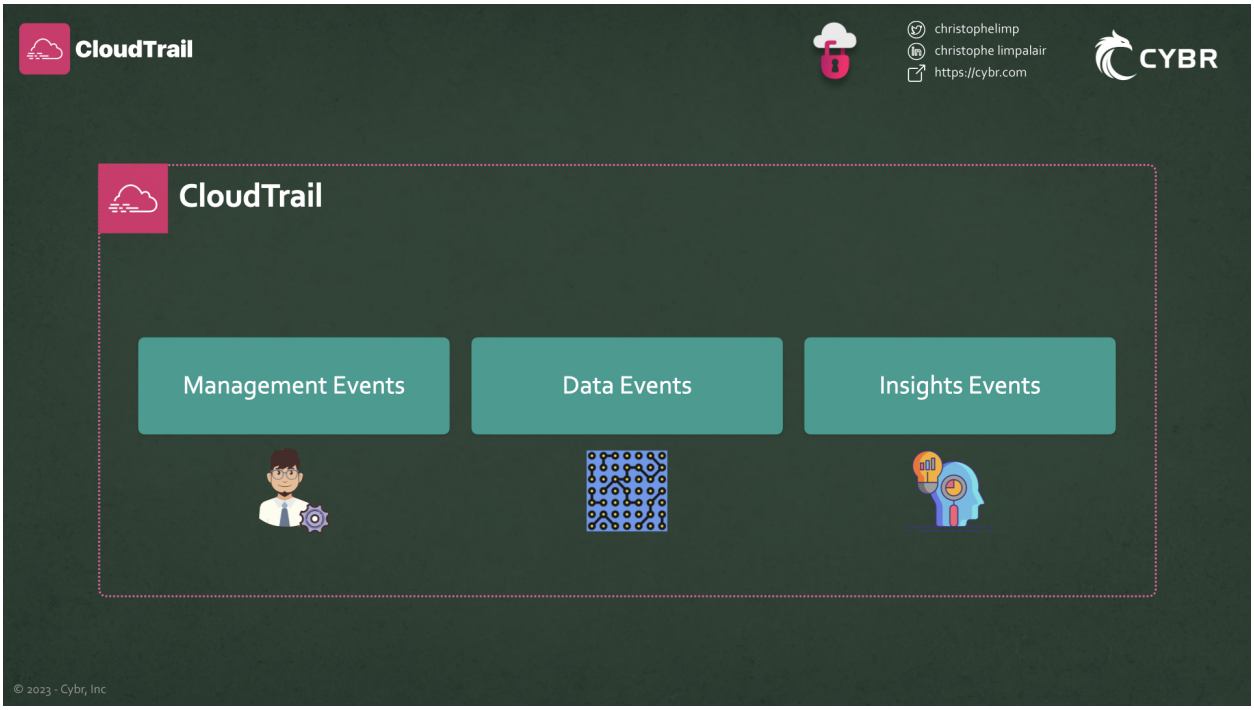
All that to say: I have years of experience working in AWS and building as well as securing production environments, and I'm now working on sharing my experiences in this (and other) courses to help you secure your own AWS resources.

For more details about my background, check out my LinkedIn profile and feel free to connect!

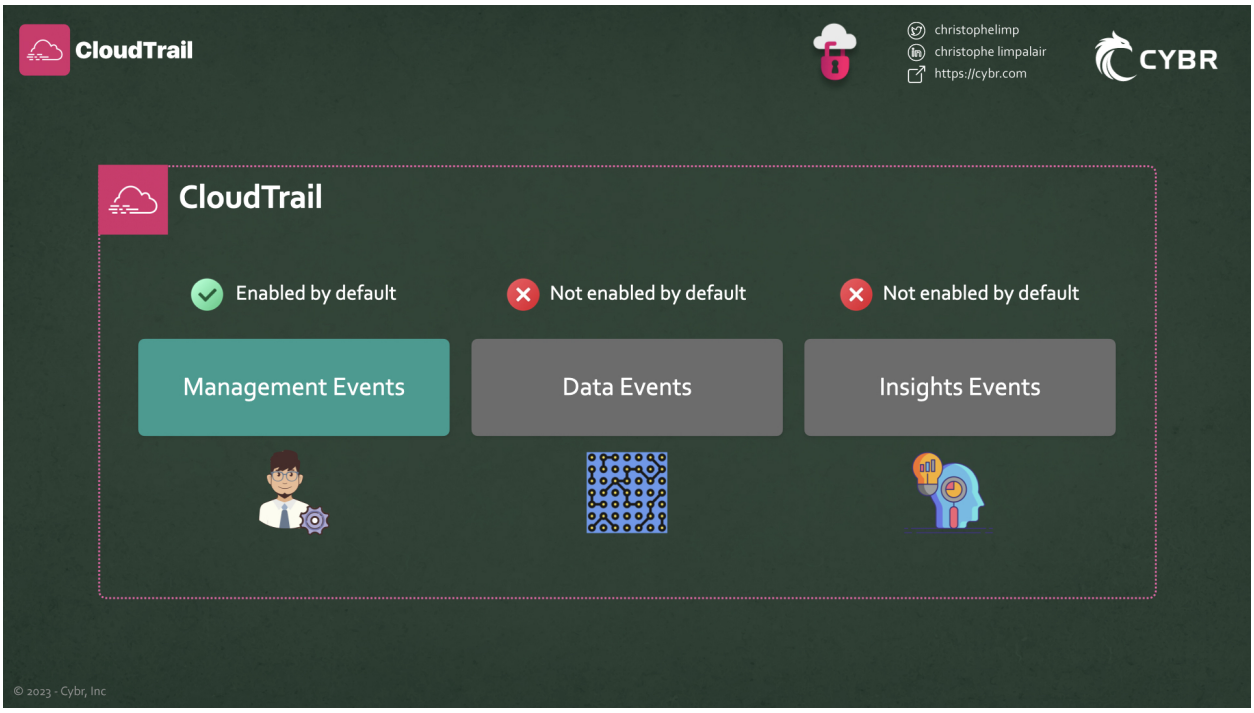
About the syllabus

This course is designed for anyone, regardless of current skill level, to be able to walk away with a thorough understanding of what CloudTrail is, how it can be used, and how you should configure it to get observability in your AWS environments depending on your specific use case.

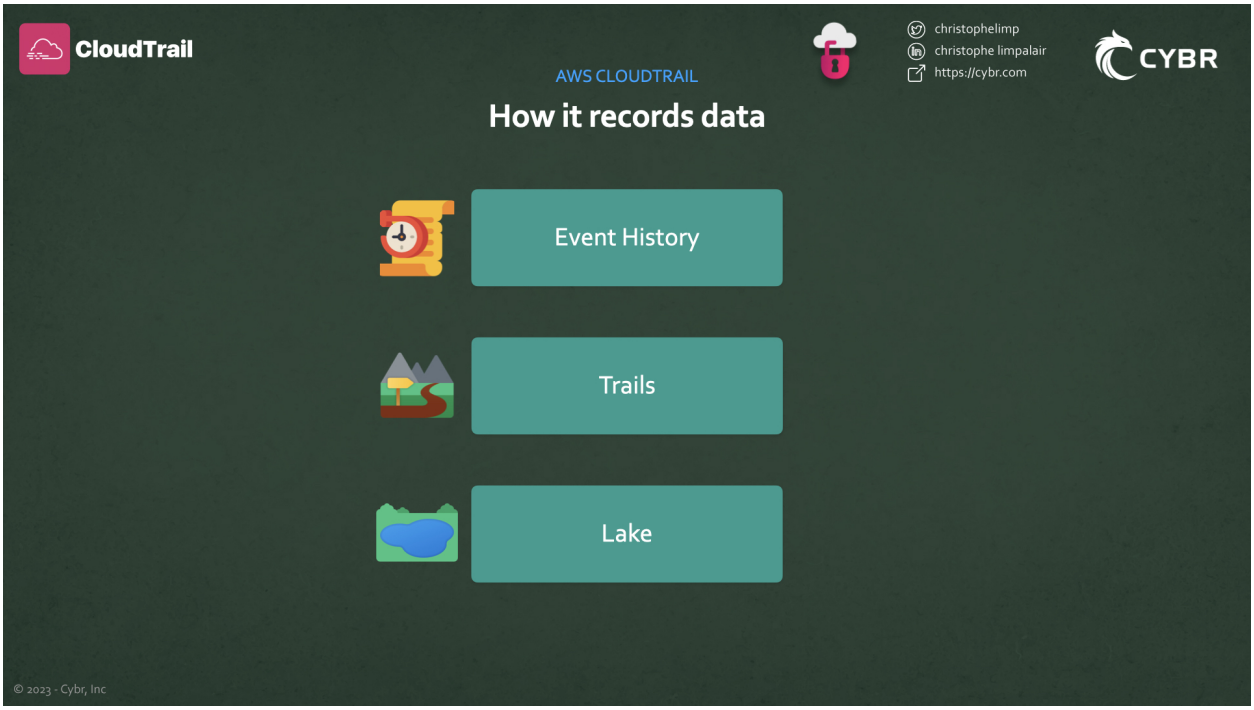
We start off by explaining what data CloudTrail is able to collect and log depending on how it's configured. There is a common misunderstanding that I've found from working with others in the industry, regarding the differences between logging Management, Data, or Insights Events.



When you create an AWS account, by default, only Management Events get collected, and this can be a problem that we'll explore in detail.



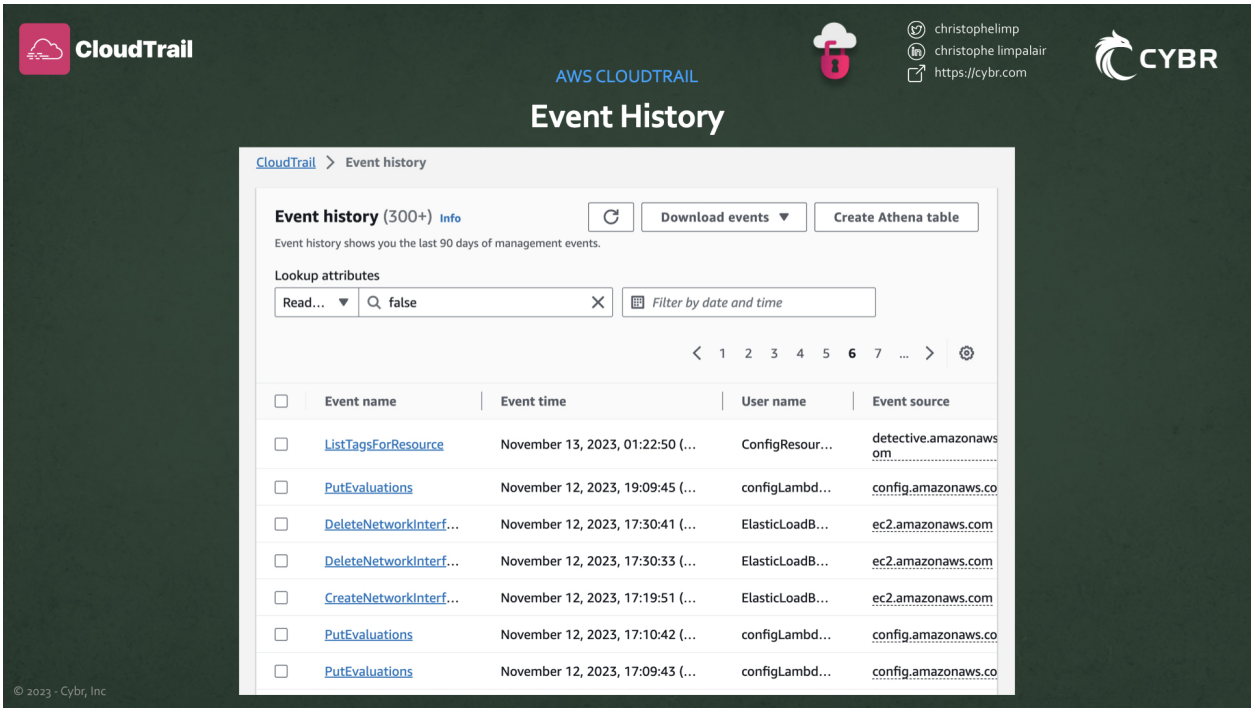
In addition, CloudTrail offers 3 primary ways of recording data, and each of those ways has pros and cons that are important to understand, and so we'll cover those topics.

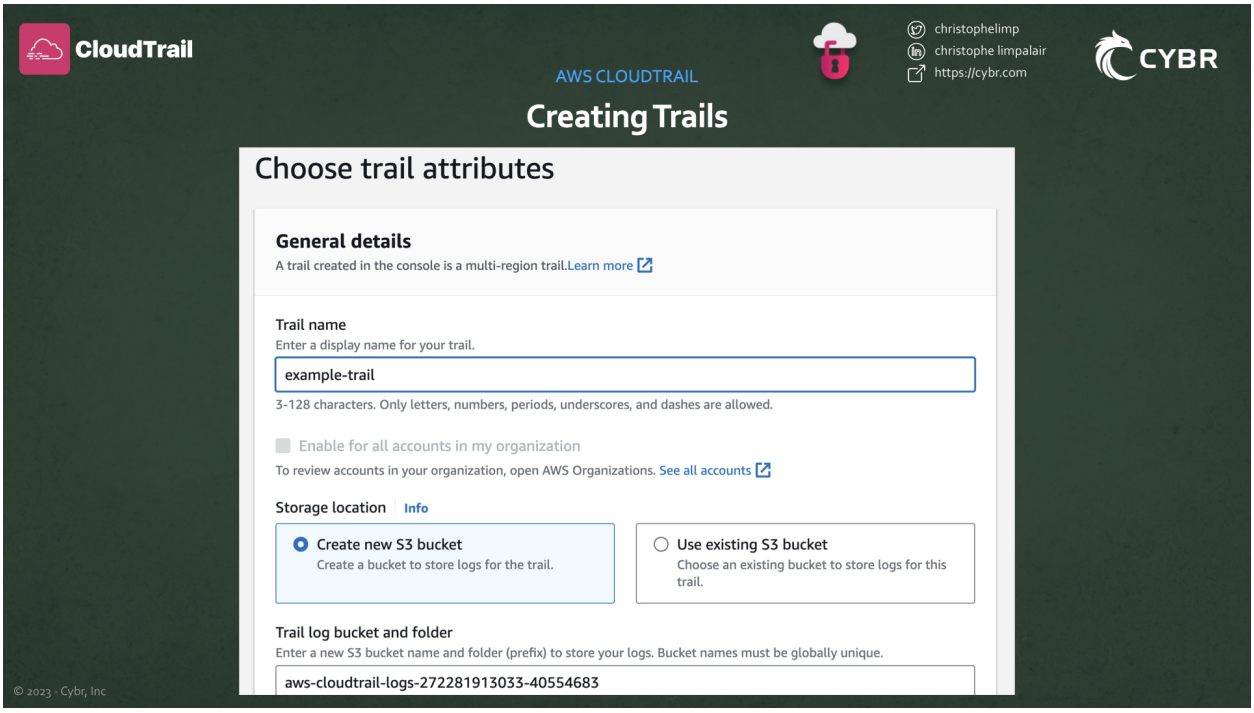


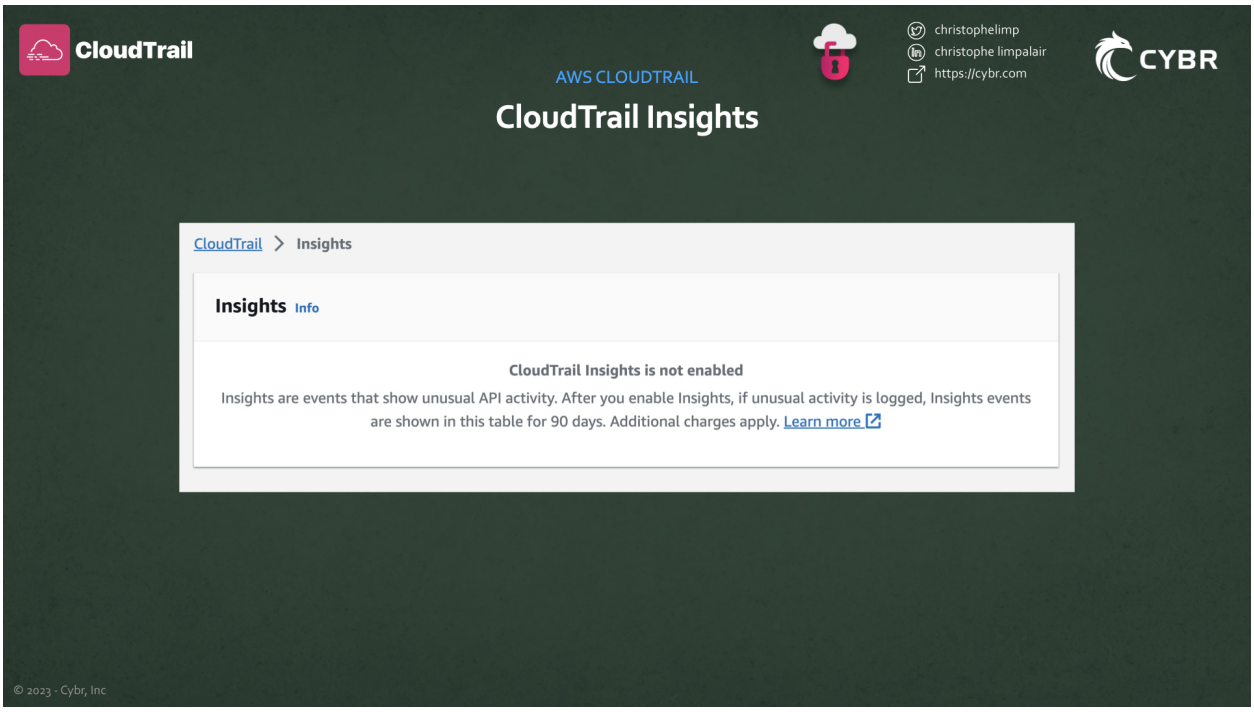
Then, we jump into the AWS console to learn how to:

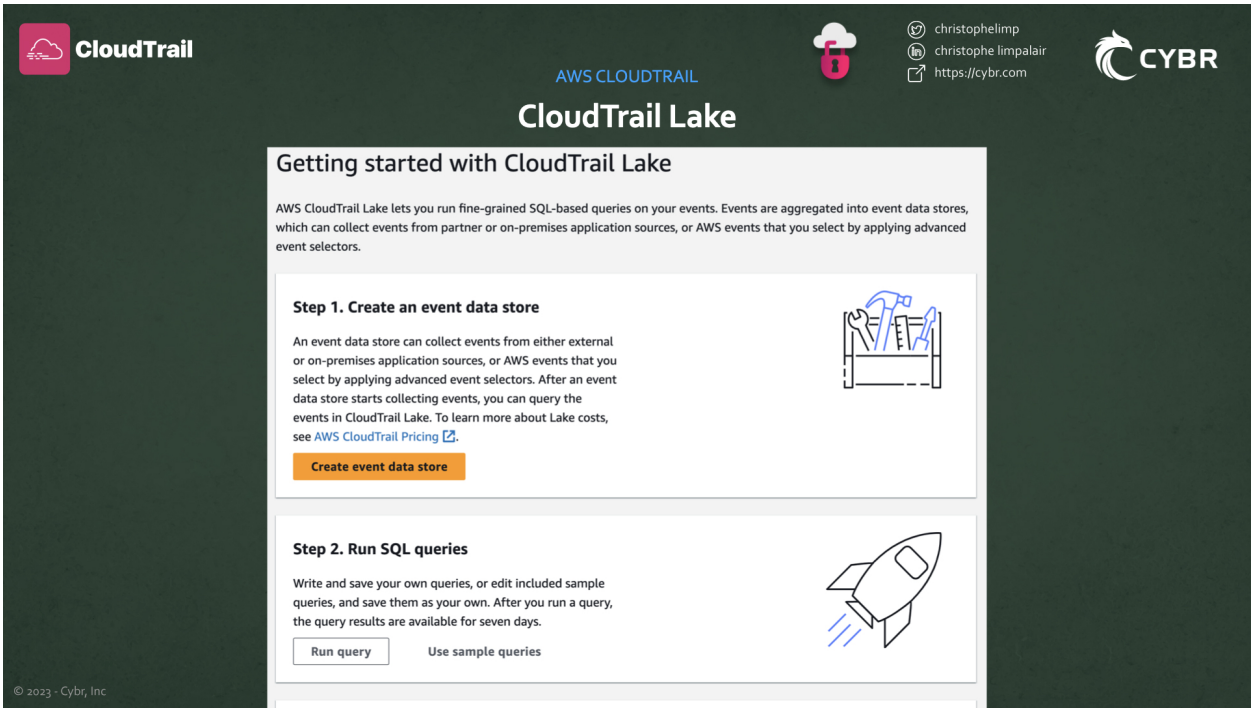
- Work with Event history

- Create our first trail
- Work with trails
- Work with pushing logs to CloudWatch for analytics, monitoring, and alerting
- Work with CloudTrail Insights to detect unusual activity
- Use CloudTrail Lake for auditing, security investigation, and operational troubleshooting




CloudTrail
christophelimp
christophe limpalair
https://cybr.com
CYBR
AWS CLOUDTRAIL
Creating Trails
Choose trail attributes
General details
A trail created in the console is a multi-region trail.Learn more
Trail name
Enter a display name for your trail.
example-trail
3-128 characters. Only letters, numbers, periods, underscores, and dashes are allowed.
Enable for all accounts in my organization
To review accounts in your organization, open AWS Organizations. See all accounts
Storage location Info
Create new S3 bucket
Create a bucket to store logs for the trail.
Use existing S3 bucket
Choose an existing bucket to store logs for this trail.
Trail log bucket and folder
Enter a new S3 bucket name and folder (prefix) to store your logs. Bucket names must be globally unique.
aws-cloudtrail-logs-272281913033-40554683
© 2023 - Cybr, Inc


CloudTrail
christophelimp
christophe limpalair
https://cybr.com
CYBR
AWS CLOUDTRAIL
CloudTrail Insights
CloudTrail > Insights
Insights Info
CloudTrail Insights is not enabled
Insights are events that show unusual API activity. After you enable Insights, if unusual activity is logged, Insights events are shown in this table for 90 days. Additional charges apply. Learn more
© 2023 - Cybr, Inc



We'll then wrap up the course by explaining how to secure our CloudTrail deployments and log files, by using IAM, log file integrity, encryption, and a checklist of security best practices.

By the end of this course, you will feel confident deploying CloudTrail for production environments, and you'll be ready to learn more advanced topics like threat hunting and investigating security

events, which we will teach in a separate course.

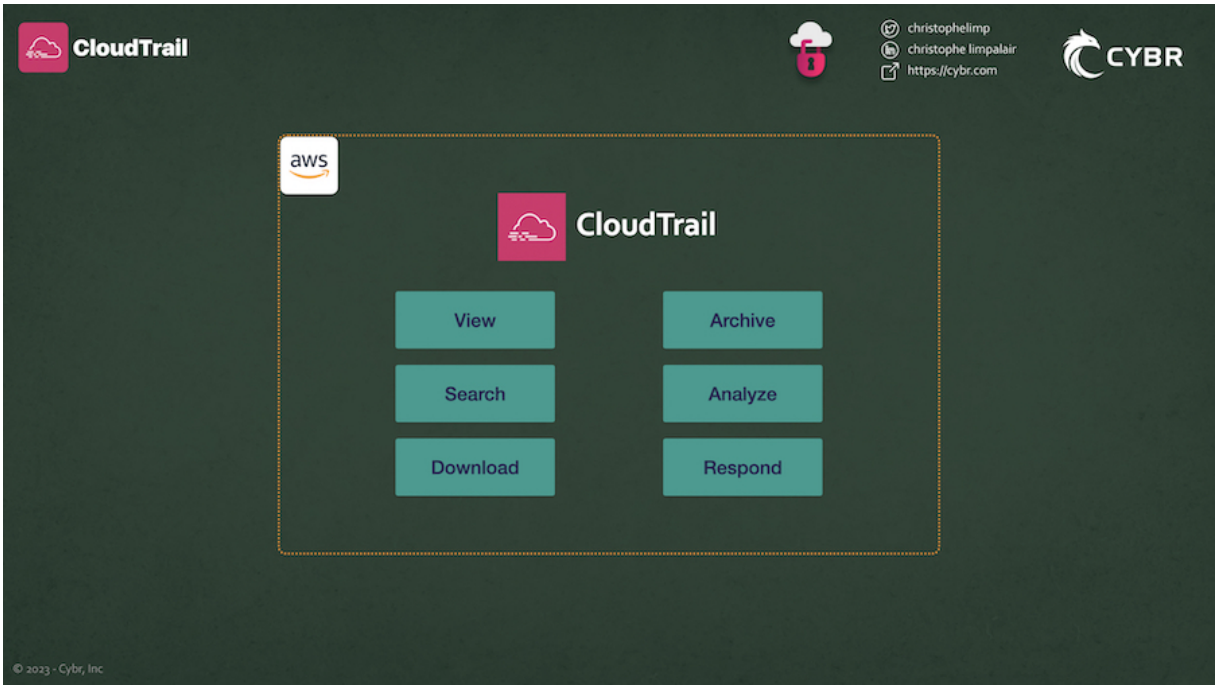
Conclusion

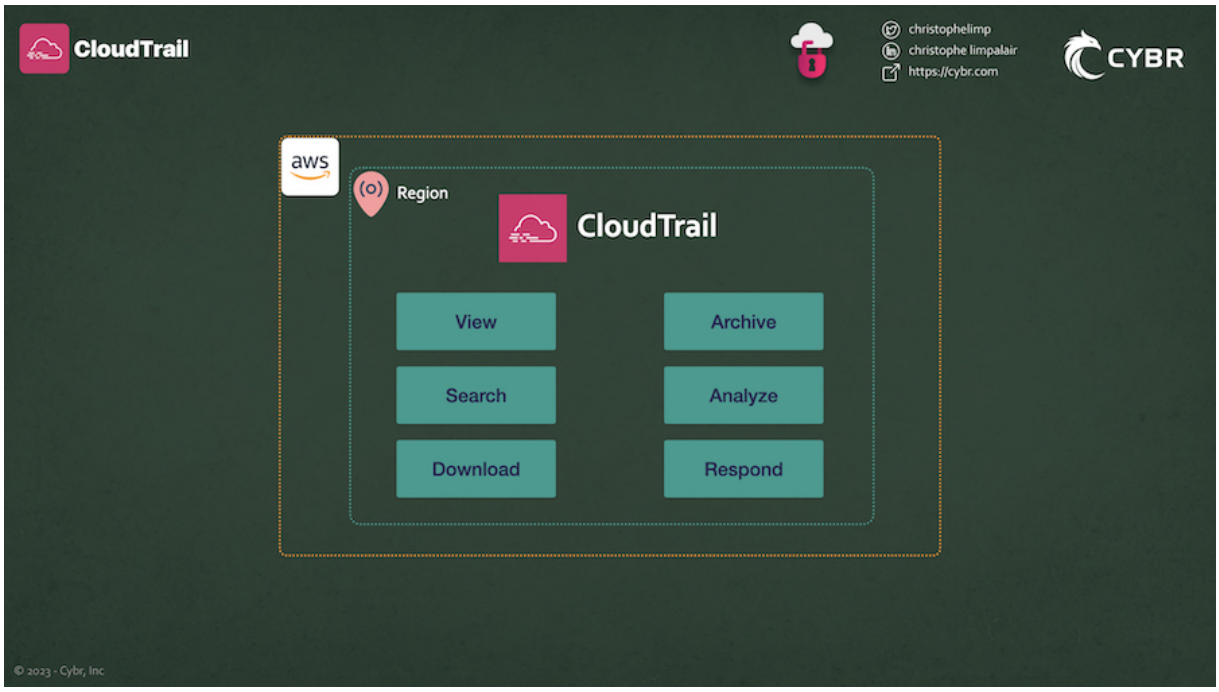
If you have any questions about this course, please reach out in our community. But otherwise, I hope you're excited to get started and I'll see you in the course!

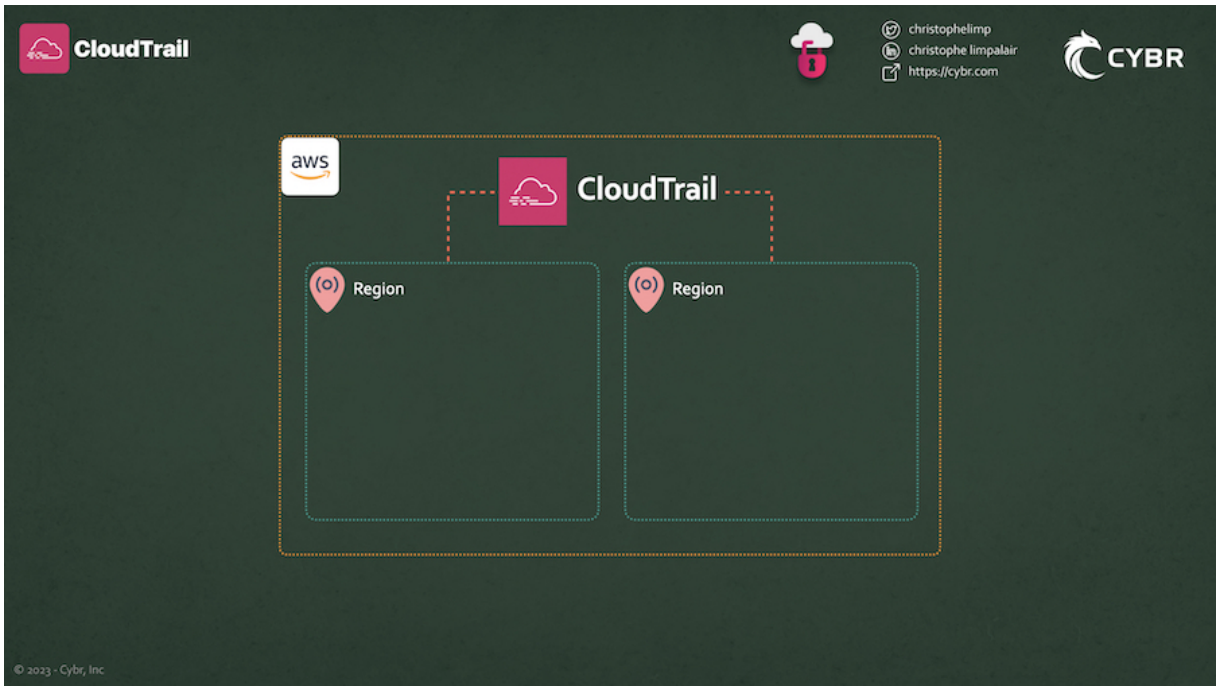
What is CloudTrail?

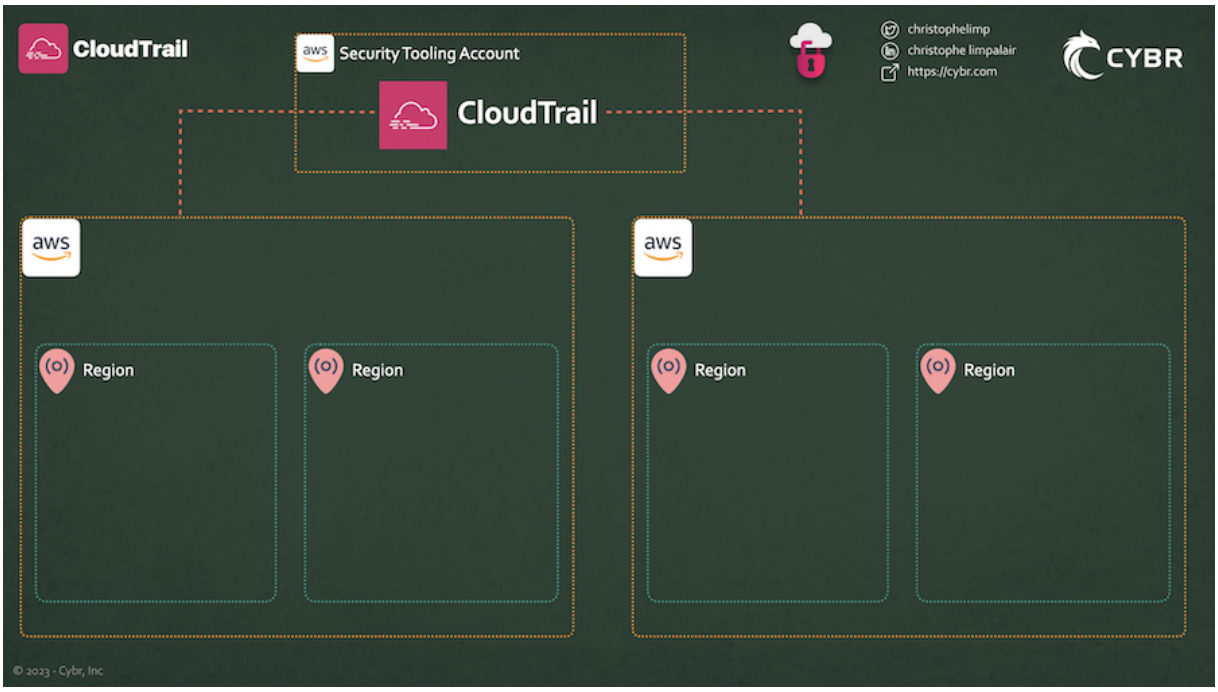
Welcome to Cybr's Getting Started with CloudTrail course! To get things started, let's talk about what this service offers, and why it's so important to understand.

With CloudTrail, you can view, search, download, archive, analyze, and respond to account activity across either a single Region in your account, multiple Regions, or even multiple accounts.




CloudTrail
christophelimp
christophe limpalair
https://cybr.com
CYBR
aws
Region
CloudTrail
View
Archive
Search
Analyze
Download
Respond
© 2023 - Cybr, Inc


CloudTrail
christophelimp
christophe limpalair
https://cybr.com
CYBR
aws
CloudTrail
Region
Region
© 2023 - Cybr, Inc

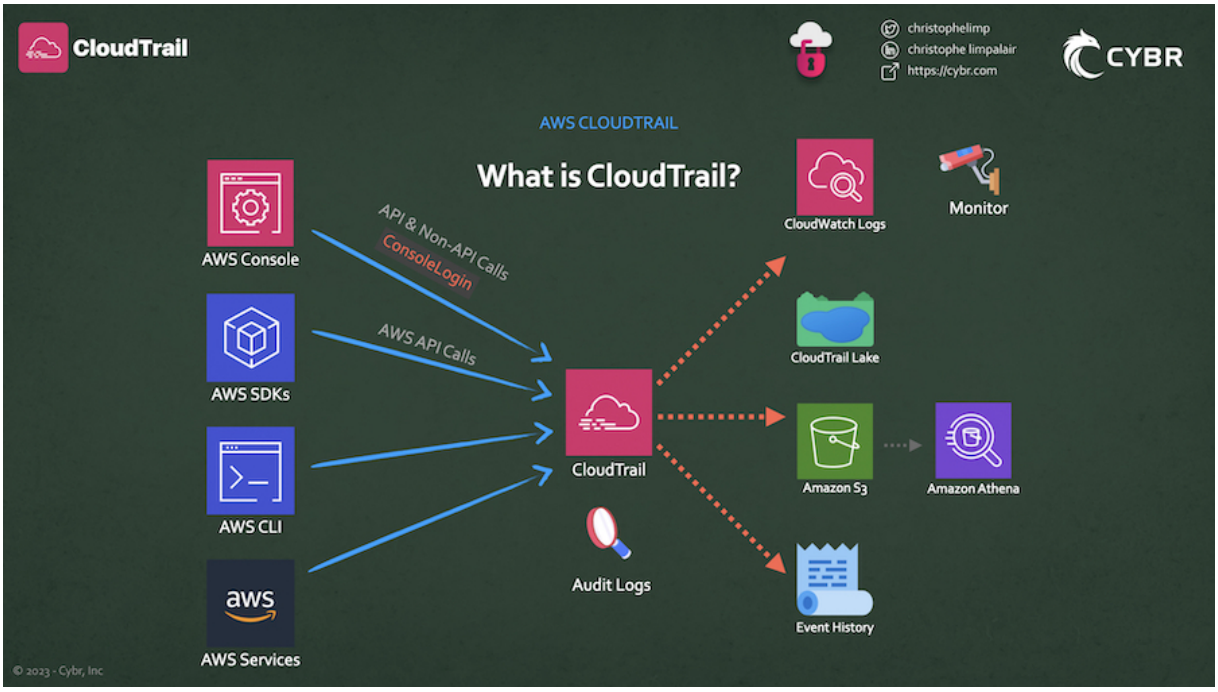


It can be configured to collect logs from multiple sources, giving you a complete history of events and API calls made within your AWS account by these sources:

- Console
- SDK
- CLI
- AWS Services

This includes AWS API calls, and some additional non-API events like `ConsoleLogins`, which is practically all of the actions across your AWS infrastructure.

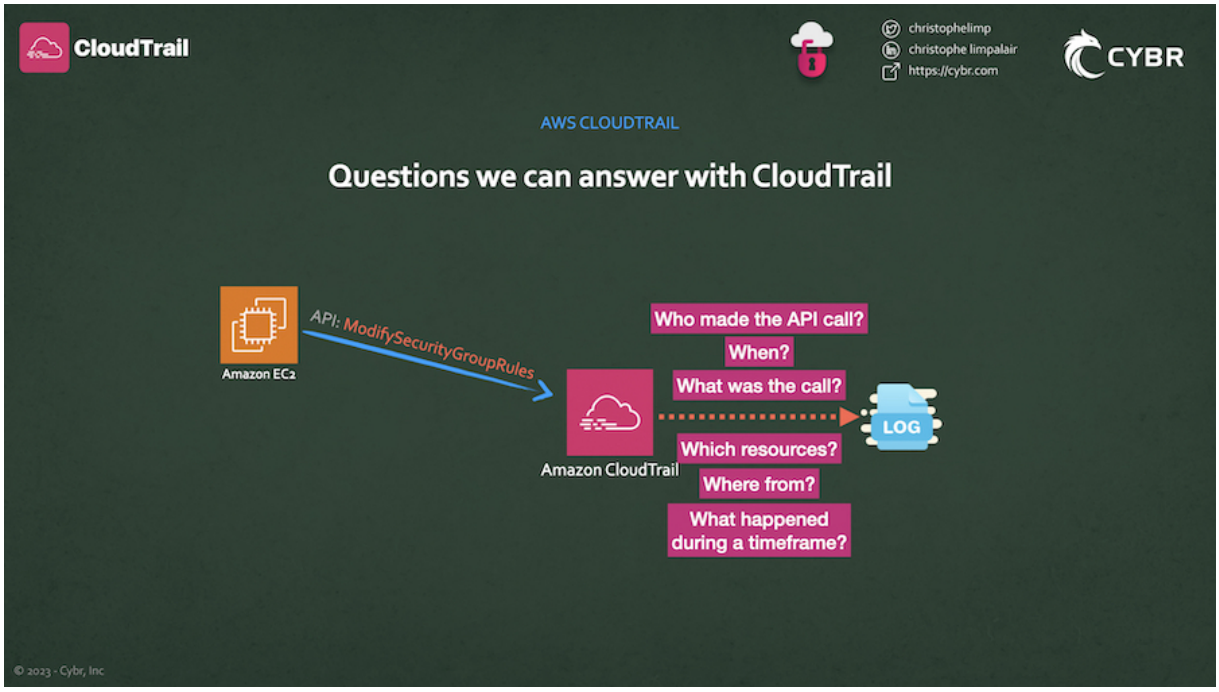
You can then use those logs to audit your AWS account and and it provides a very useful source for monitoring simple and large or complex AWS environments for suspicious or potentially dangerous actions.



If something weird is going on in your account, especially if it's security related (but also operational issues), CloudTrail's data should probably be one of the first — if not the first — places that you investigate, and we'll talk more about how to do that later in the course.

It's a service that helps you answer important questions, like:

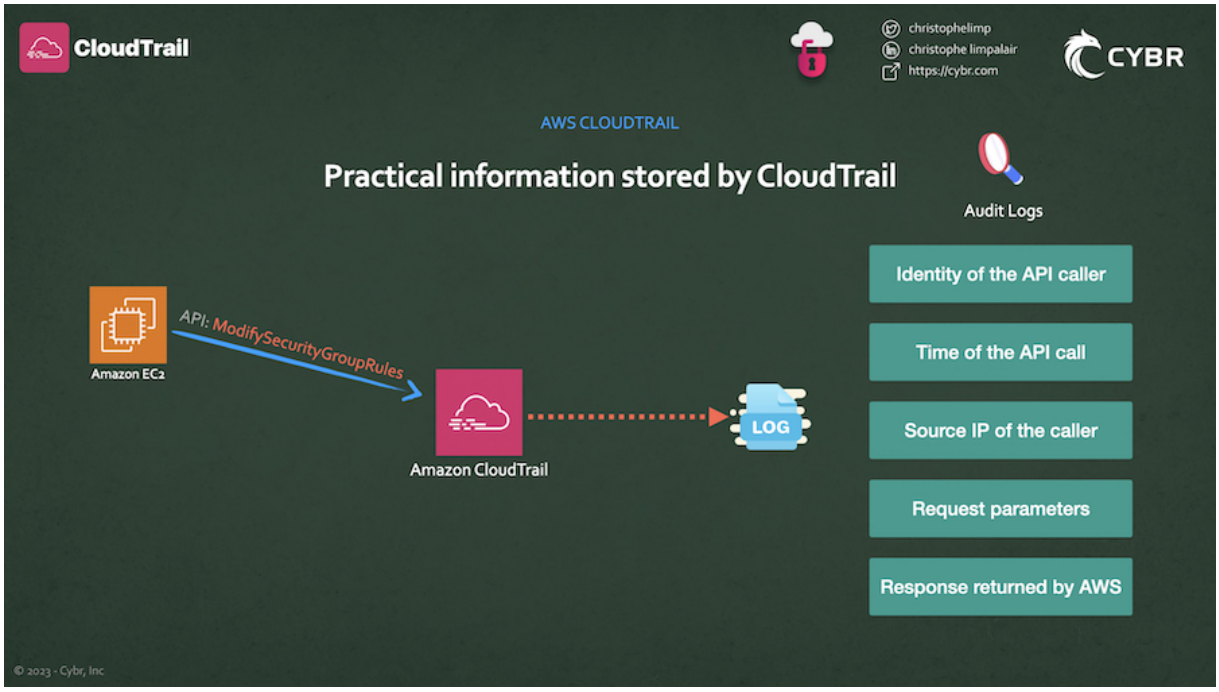
- "Who took this action and when?"
- "What resources did they access or modify?"
- "What events happened during a certain timeframe?"
- Etc…



For example, if I make a change to an EC2 instance, even if that change was made through the console, it's still making calls to the AWS API, which means CloudTrail would see it and log it.

You can find out information like the:

- Identity of the API caller
- Time of the API call
- Source IP address of the API caller
- Request parameters
- Response elements returned by the AWS service



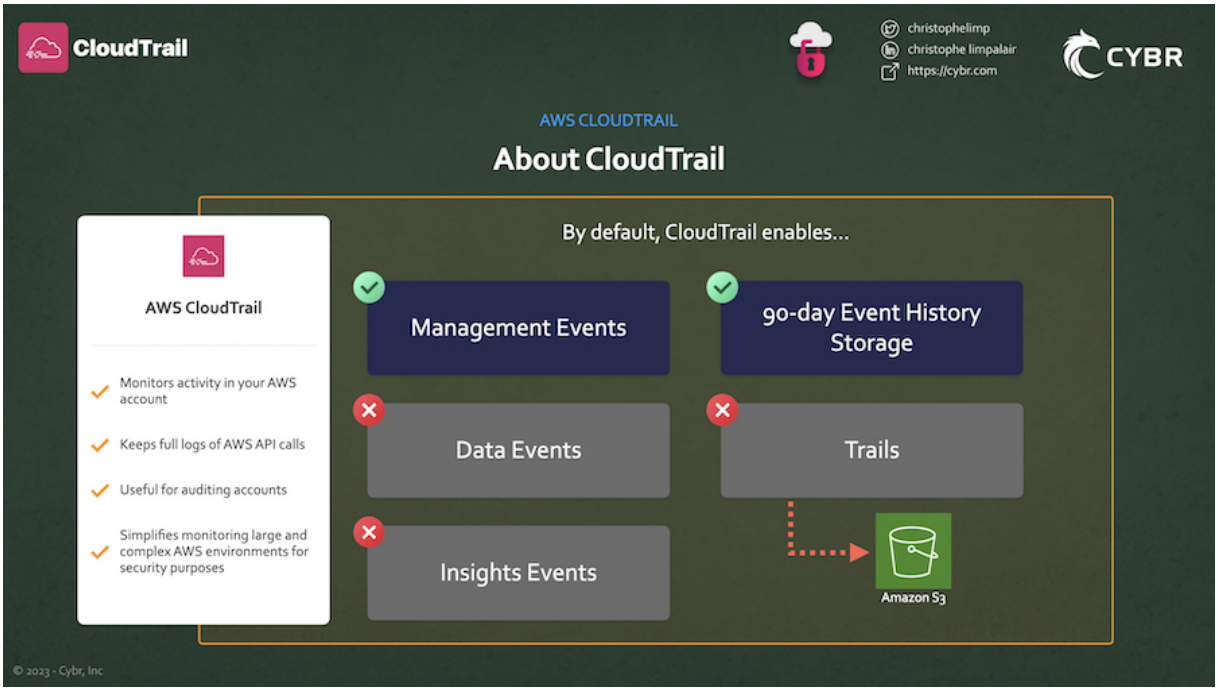
However, it's important to understand that all of the functionality we just mentioned is *not* enabled by default!

Whenever you create a new AWS account, a limited version of CloudTrail is automatically enabled on your behalf. That limited version only logs Management Events, and not Data Events or Insights Events.

We'll learn more about what this means in other lessons, because this is very important to understand.

By default, CloudTrail also doesn't get enabled with any trails. Instead, the Management Events get stored for up to 90 days in what's called the Event History.

There are ways of storing data for longer than that, which we will also be exploring, but as one of the first steps, you will want to generate CloudTrail trails and I'll show you how to do that.



Example events to keep track of

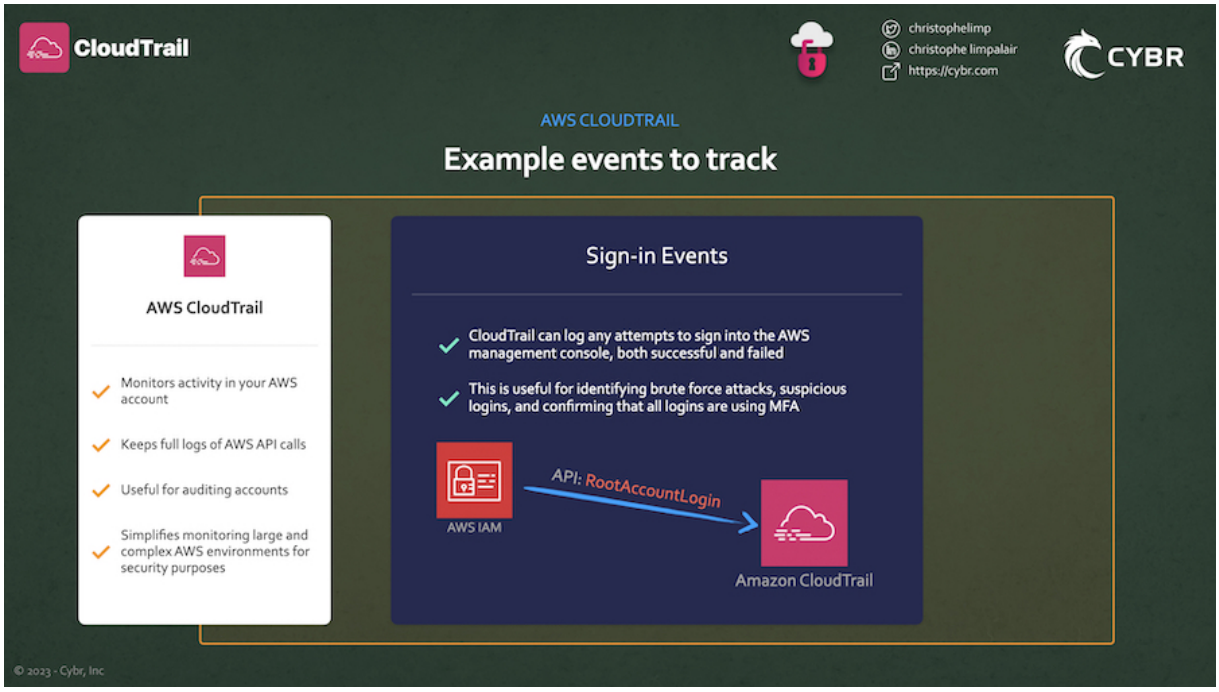
Let’s take a look at some actual examples of what CloudTrail can log, and why it’s important.

Sign-in events

One of the most important events that you need to keep track of is console sign-in events. CloudTrail can log any attempts to sign into the AWS management console, both successful and failed.

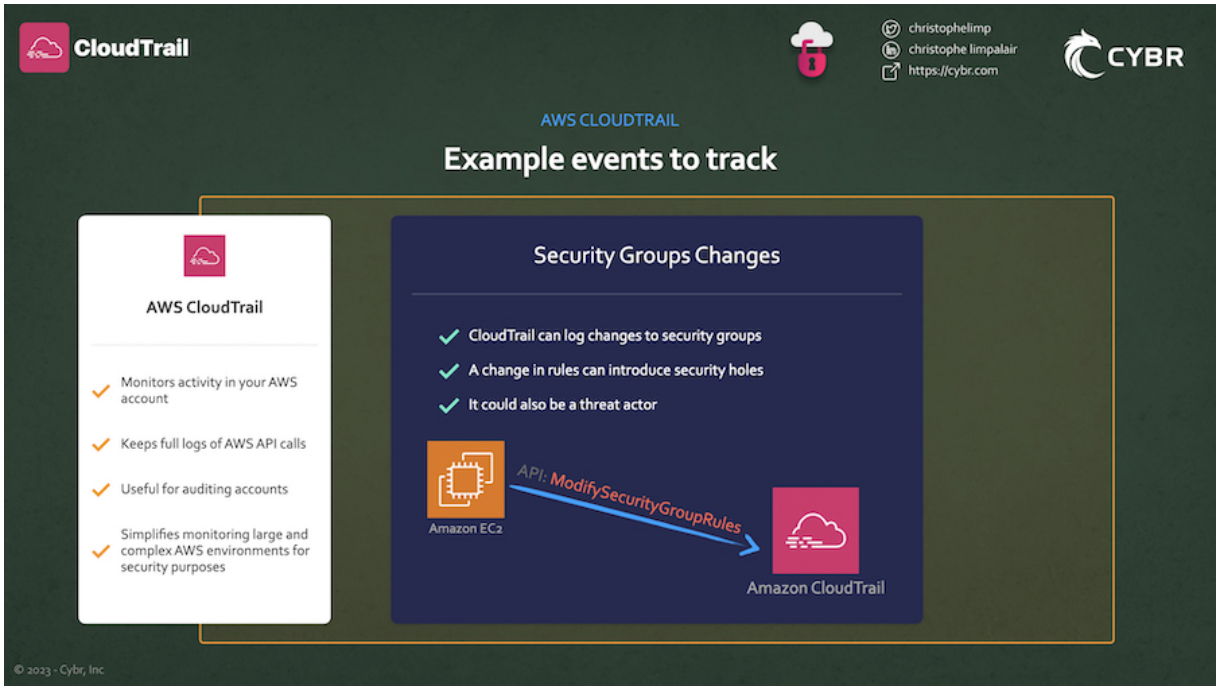
This way you can identify brute-force attempts, or suspicious logins that took place after hours, and confirm that all logins are using multifactor authentication. Remember that your console is your most critical asset. Anyone with access to your console can potentially cause a lot of damage.

You can also set up monitoring and alerting for any time someone logs in using your root account. Because root accounts should almost never be used, if someone is using one, you should get notified, and you can use CloudTrail’s logs to do that.



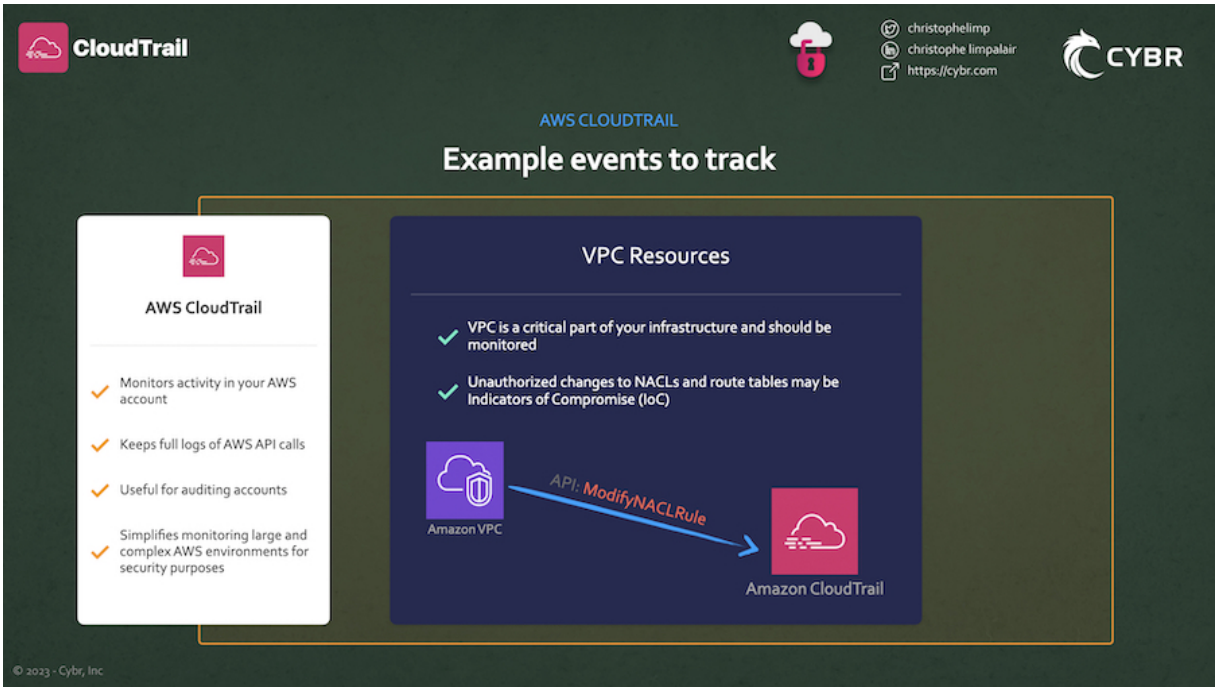
Security Groups changes

CloudTrail also has the ability to log changes to security groups, as another example. A change in a security group can introduce, even by accident, a security hole in your AWS environment. You should monitor such changes closely and ensure that they occur as part of a change management process.



VPC resources

Apart from security groups, it is important to keep track of several changes that may occur in your VPCs and can potentially have a security impact. Unauthorized changes to NACLs and route tables may be indicators of compromise (IoCs). Even if made on purpose, such changes may have a negative impact on the security posture of your AWS environment, and you should definitely keep an eye on them.



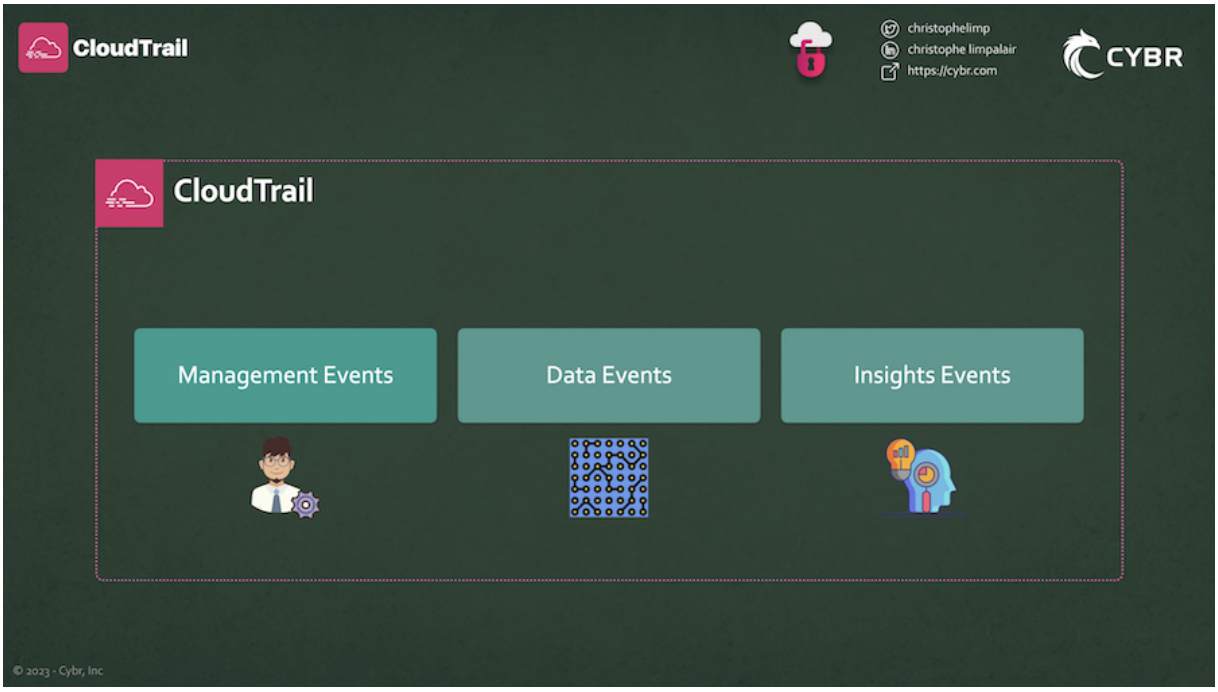
Conclusion

These are just a handful of practical examples, but as we will see in this course, there are many more applications.

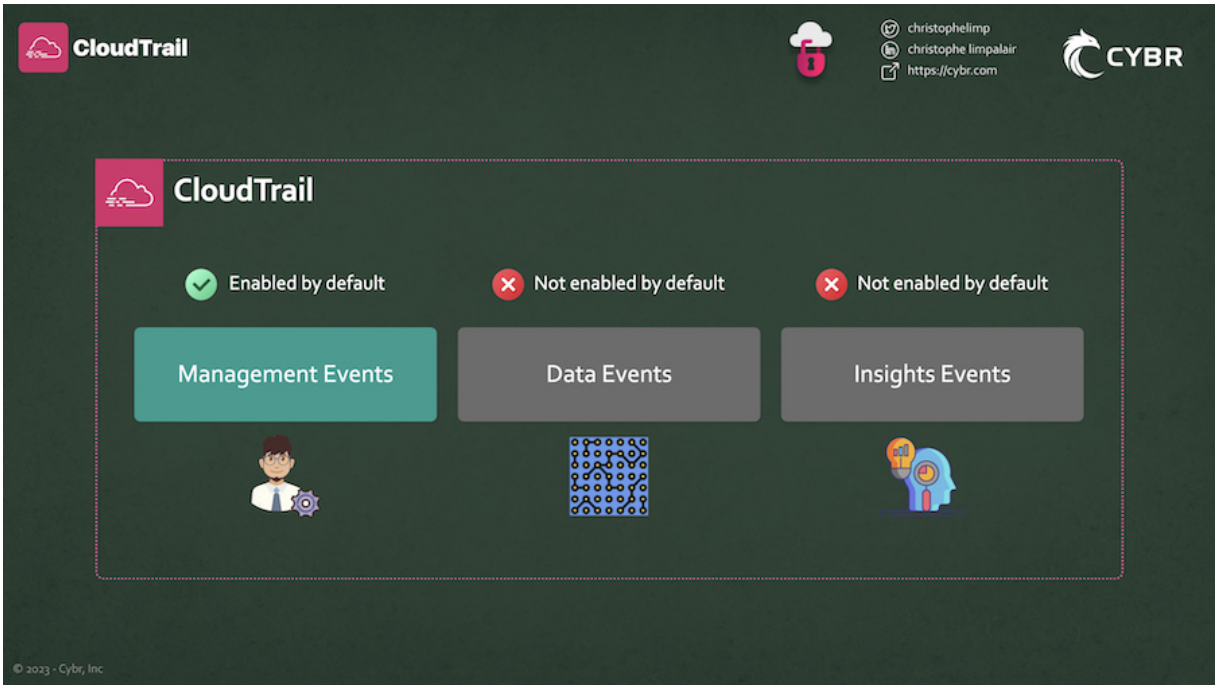
Now that we've gotten an overview of what CloudTrail is, how it works, and when it's useful, we're ready to move on to the next lesson!

Management, Data, and Insights Events

There's a very big difference between logging Management Events, Data Events, and Insights Events.



Management Events logging is what's enabled by default when you create an AWS account. It's helpful, but a lot of people misunderstand and believe that it logs more than it actually does.



If they don't understand this and they don't enable Data Events, and a security incident happens, they may be left completely blind as to what's going on because they're not getting the data they needed.

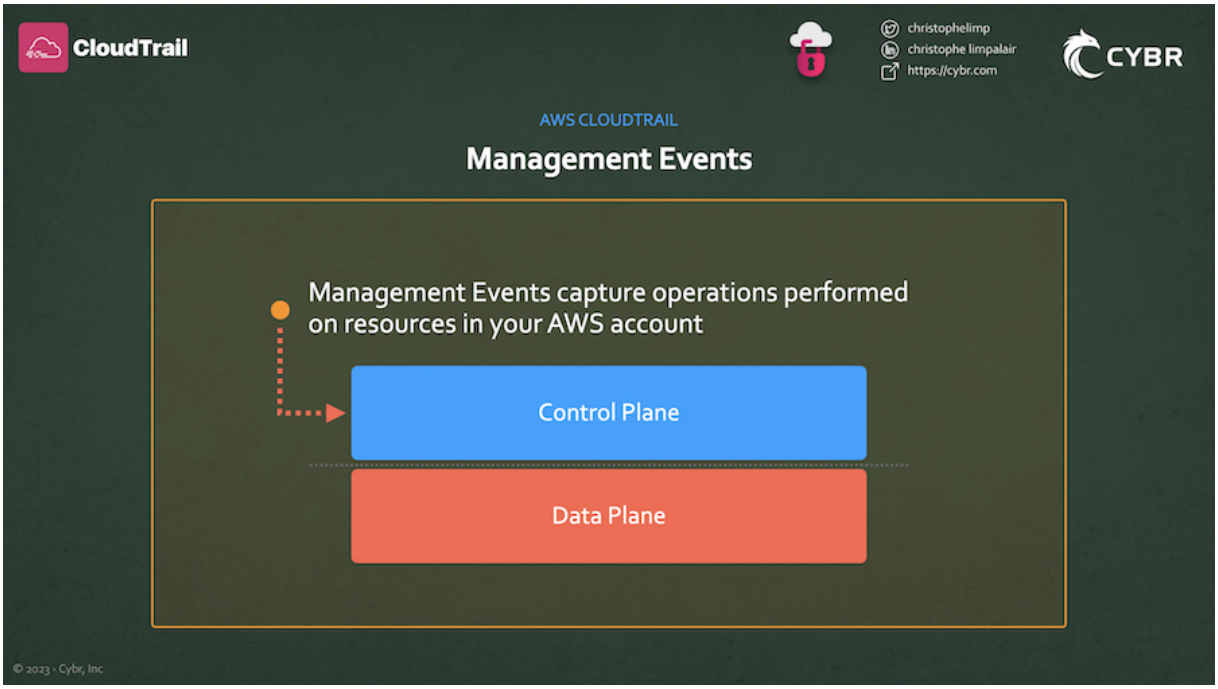
Even if they do go in and enable additional logging because they think an incident occurred, they won't be able to see back in time. They will only see what's happening now and in the future. So unless the attack is still ongoing, they probably won't get any useful information they can use to investigate the incident, unless they're able to get it from some other service.

That's why this lesson is so important and we need to cover this topic before we go in and configure CloudTrail.

Management Events

So let's start by deep diving into what Management Events are and what they cover.

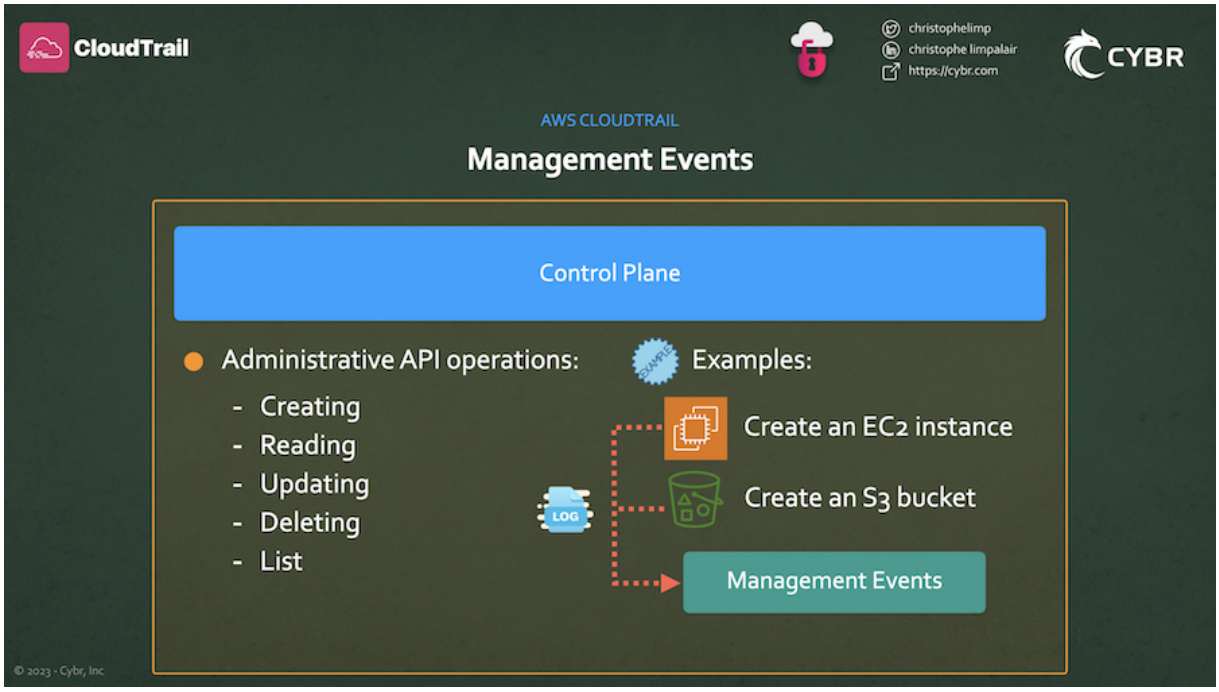
Management Events capture operations that are performed on resources in your AWS account. This is referred to as control plane operations.



A plane is a term used in IT that comes from networking, and it's used to refer to what layer the operation occurs in.

AWS separates most services into either the control plane or the data plane.

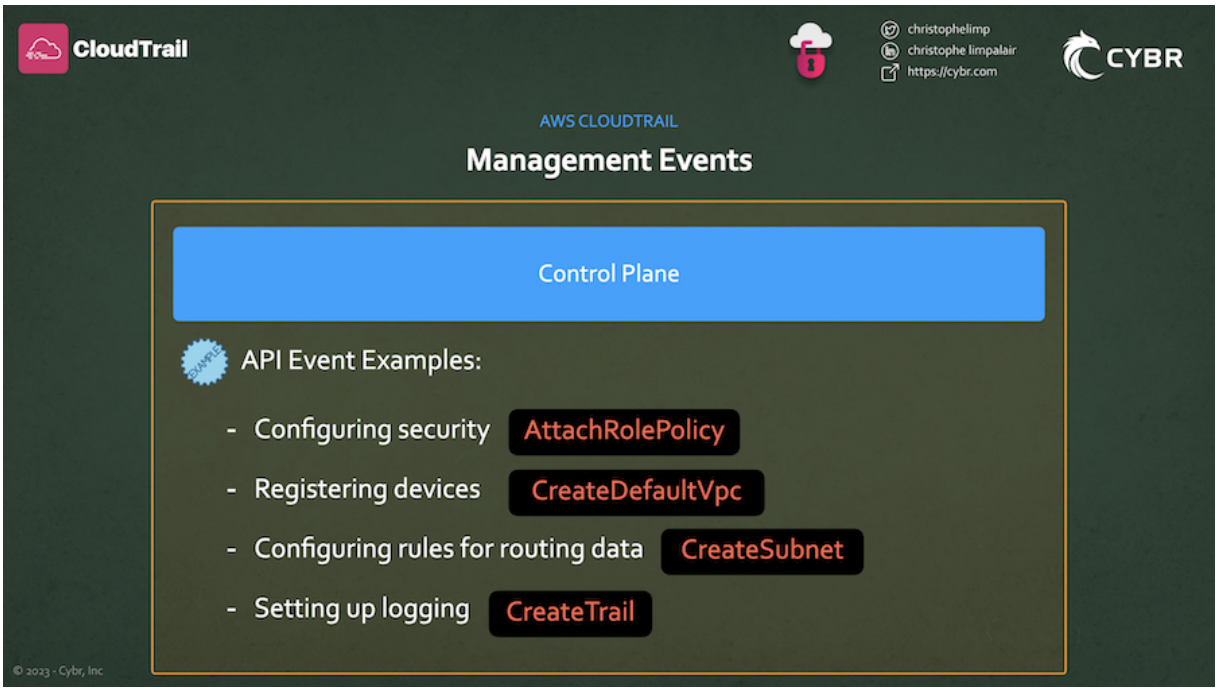
The control plane refers to administrative API operations, like creating, reading, updating, deleting, and listing resources.



For example, launching an EC2 instance or creating an S3 bucket is an action in the control plane, and so that would be logged as part of Management Events in CloudTrail.

Other examples include:

- Configuring security (for example, IAM `AttachRolePolicy` API operations)
- Registering devices (for example, Amazon EC2 `CreateDefaultVpc` API operations)
- Configuring rules for routing data (for example, Amazon EC2 `CreateSubnet` API operations)
- Setting up logging (for example, AWS CloudTrail `CreateTrail` API operations)

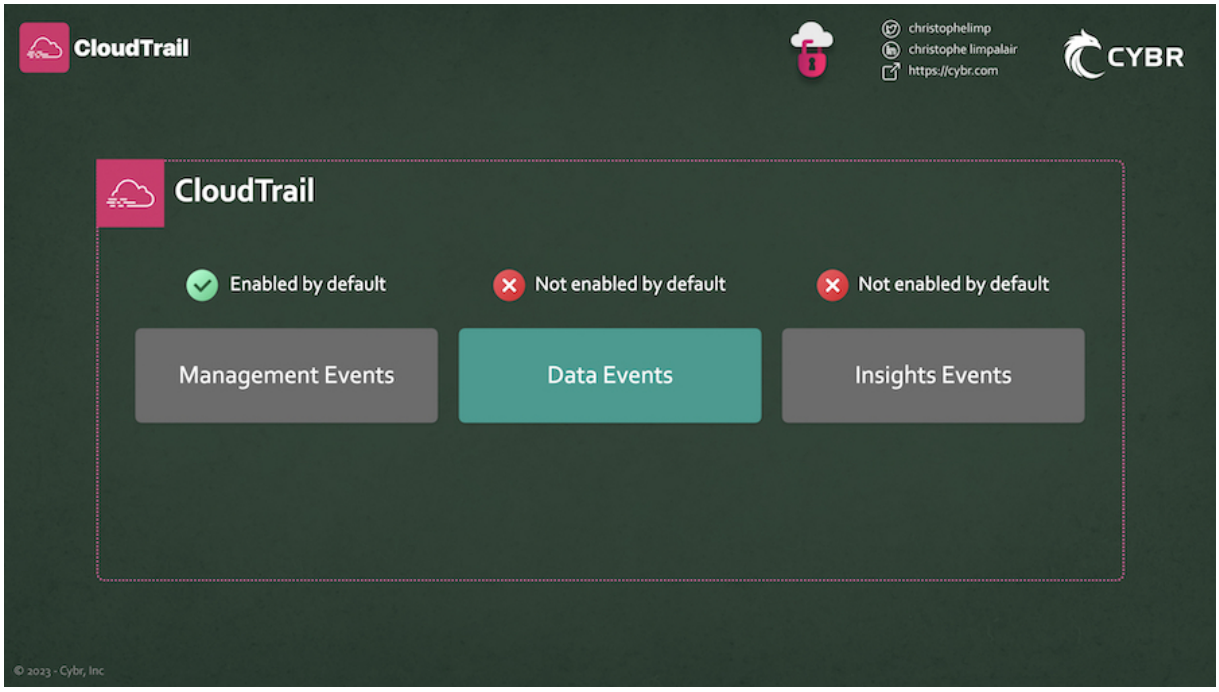


Management Events can also include non-API events in your AWS account. Non-API events include:

- AWS service events – which are events created by AWS services but not directly triggered by a request you made to a public AWS API. An example of this is when a customer managed key is automatically rotated in AWS KMS [Source]
- AWS Management Console sign-in events – examples include successfully or unsuccessfully signing in as an IAM user or as a root user, a root user changing their password or changing their MFA settings, etc… For more examples, please refer to the AWS documentation

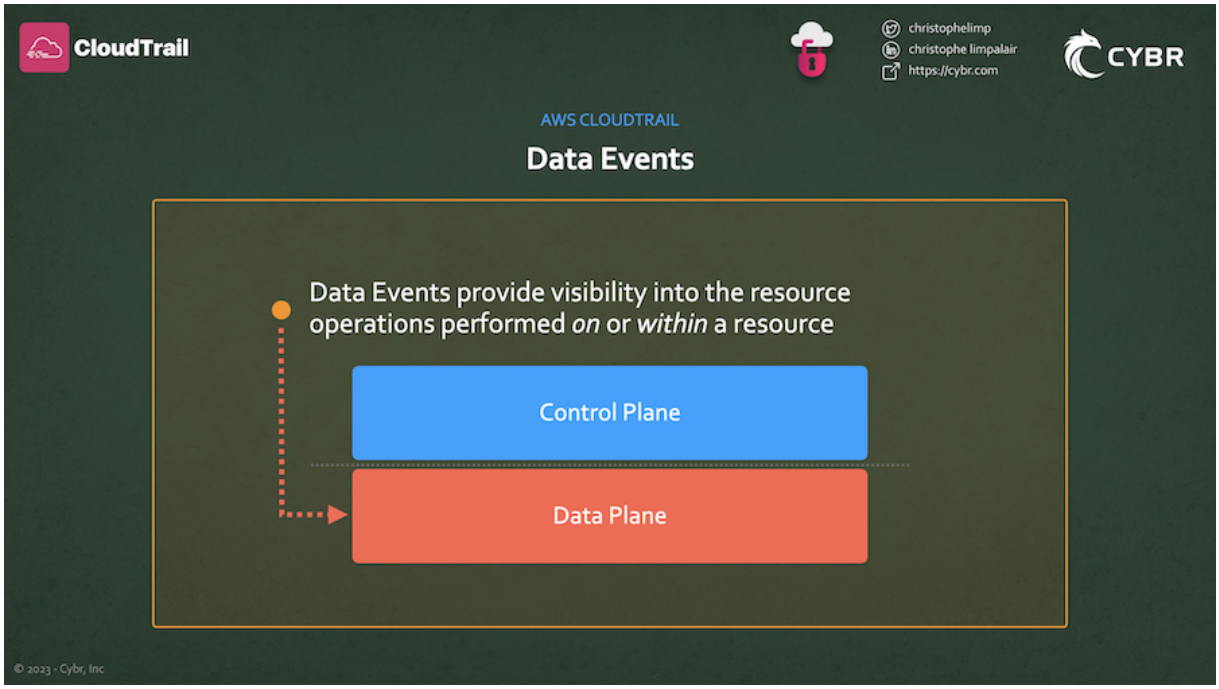
So this is all considered to be part of the control plane, and part of what gets logged in Management Events by CloudTrail.

Data Events



Data Events, on the other hand, are not enabled by default. Unless you go into the CloudTrail service and configure it to specifically log them, CloudTrail will ignore this data. The reason for this is that AWS charges more for logging this type of data, and you can view pricing information here.

Data Events provide visibility into the resource operations performed *on* or *within* a resource, which is referred to as *data plane* operations.



Because AWS can change this at any time, I'd recommend that you refer to their documentation for an up-to-date and comprehensive list of what all this covers.

Some examples include:

- Amazon DynamoDB – object-level API activity on tables like `PutItem`, `DeleteItem`, etc…
- AWS Lambda – function execution activity via the `Invoke` API
- Amazon S3 – object-level API activity like `GetObject, PutObject, DeleteObject`, etc…on buckets and objects in buckets
- Even AWS CloudTrail – with `PutAuditEvents` activity on a CloudTrail Lake channel
- and much more…

So whereas Management Event data would log S3 account-level actions [Source] like `DeleteAccountPublicAccessBlock`, `GetAccountPublicAccessBlock`, or `PutAccountPublicAccessBlock`, and it would log S3 bucket-level actions [Source] like `CreateBucket`, `DeleteBucket`, etc…

Data Events data would log object-level actions [Source] like `CopyObject`, `DeleteObject`, `GetObject`, `ListObjects`, etc…

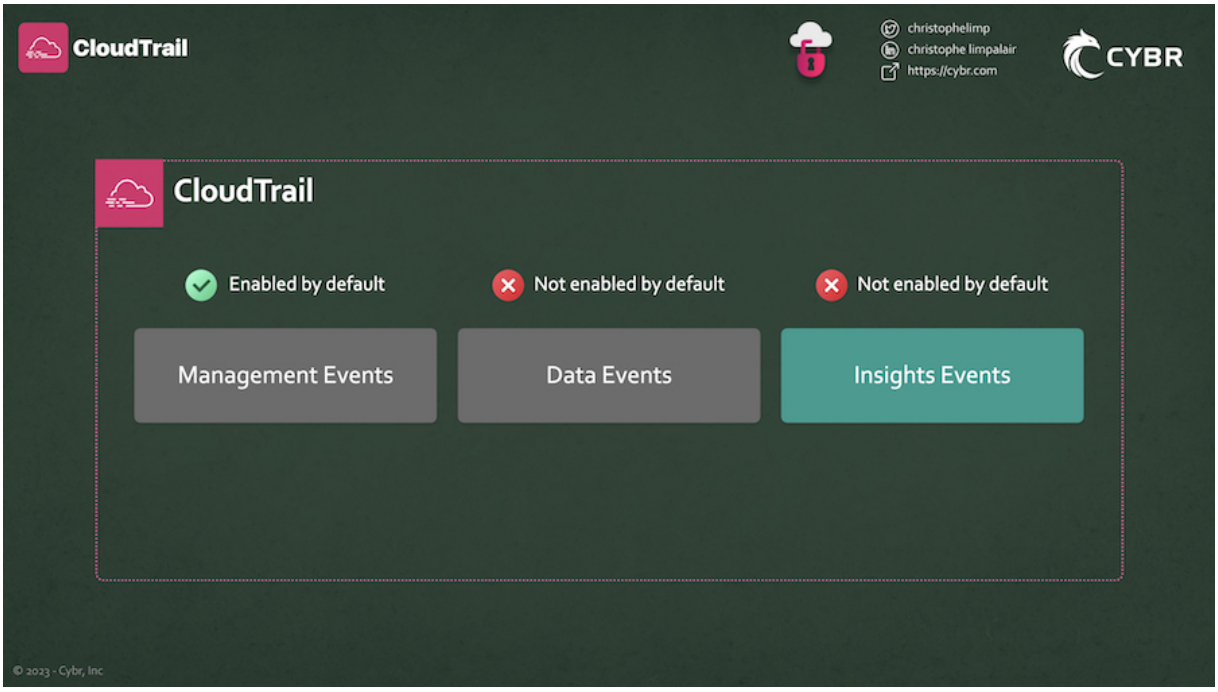
This type of data won't get logged unless you enable it. You would need to create a trail and then create an event data store, which we'll see how to do in the next section.

Now you can see why understanding this distinction is so important. If a security incident occurs that involves exfiltrated sensitive S3 data, and you only have Management Events enabled, you could completely miss the entire issue and you won't have any useful data to investigate what happened!

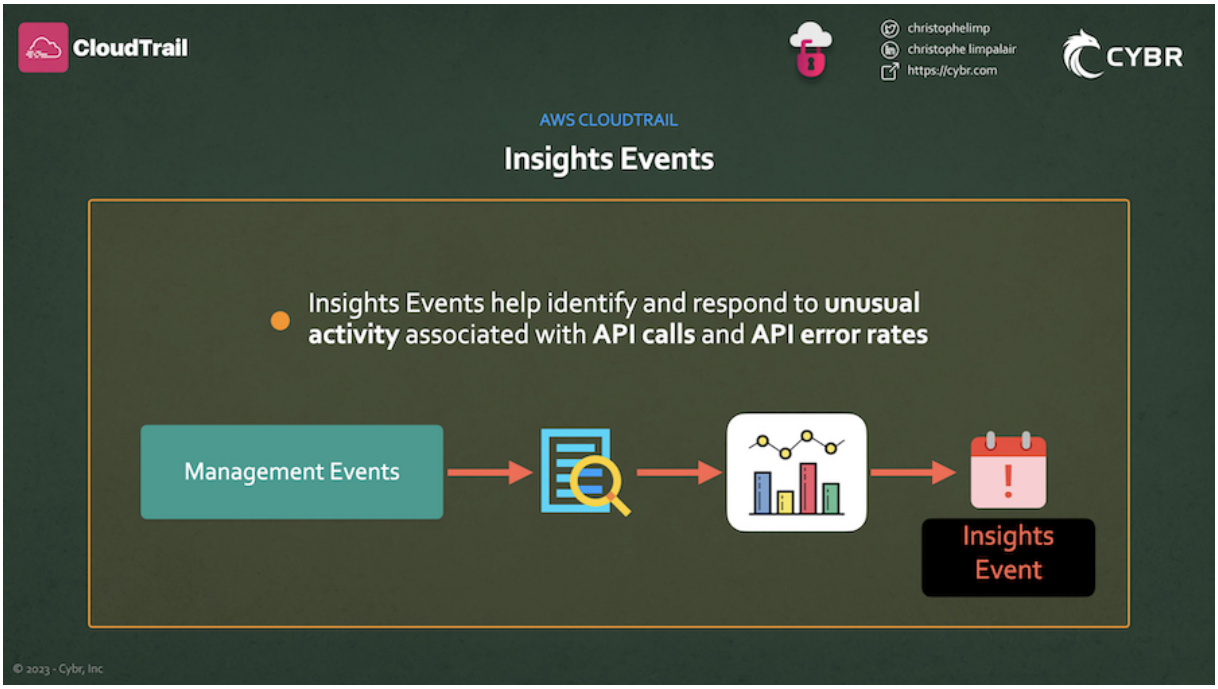
Of course, the downside to consider is cost, and that's something you or your organization has to decide on.

Insights Events

Finally, let's talk about Insights Events.



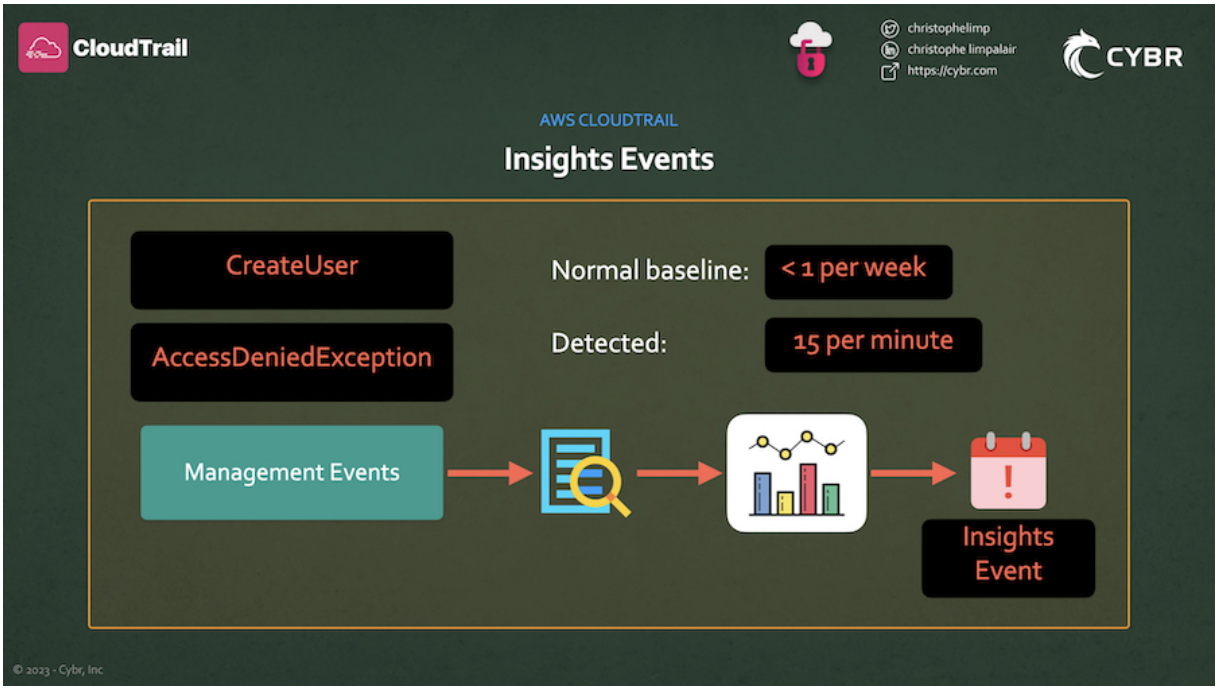
Whereas Management Events and Data Events give the ability to capture data from two different planes, Insights Events helps you identify and respond to unusual activity associated with API calls and API error rates.



It does this by constantly analyzing CloudTrail Management Events, and by creating a baseline of normal patterns, and then generating `Insights events` whenever the call volume or error rates go

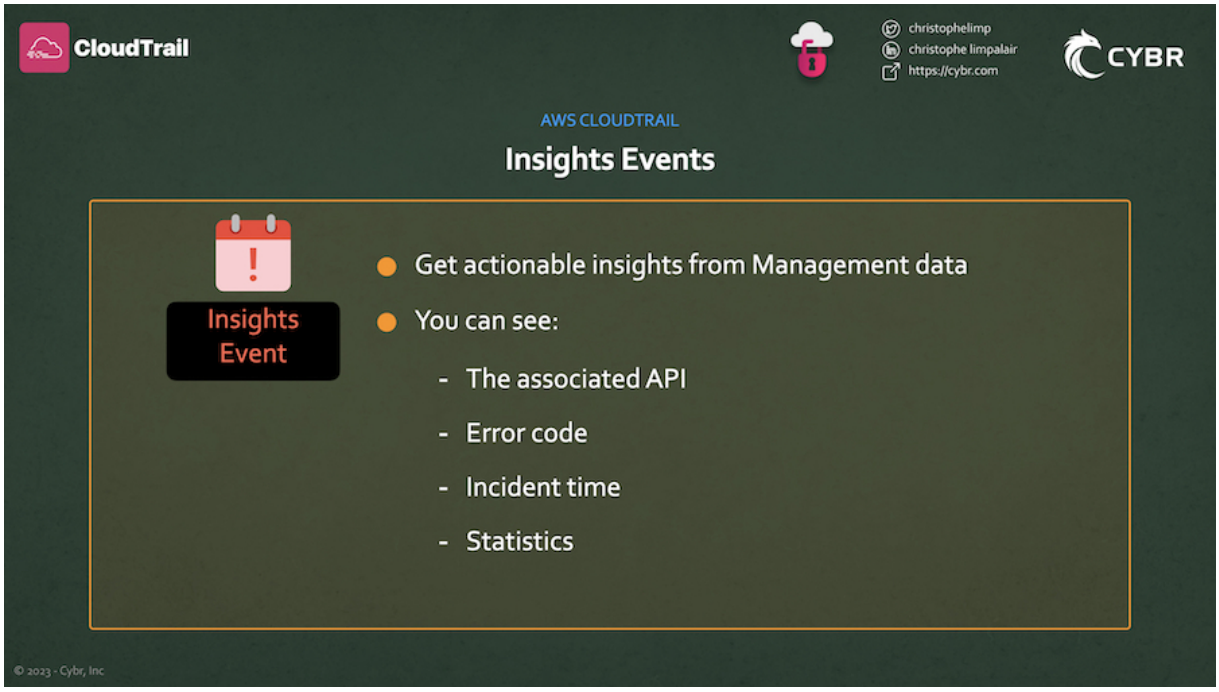
outside of that normal pattern.

For example, if your AWS account normally gets fewer than 1 `AccessDeniedException` errors in a seven-day period on the AWS IAM API call of `CreateUser`, but then you start logging an average of 15 errors per minute, an Insights Event would get logged at the start of the unusual error rate activity, and then another Insights Event would get logged to mark the end of the unusual activity.



This feature essentially focuses on giving you actionable insights based on Management data, and it provides relevant information you can use for an investigation, like:

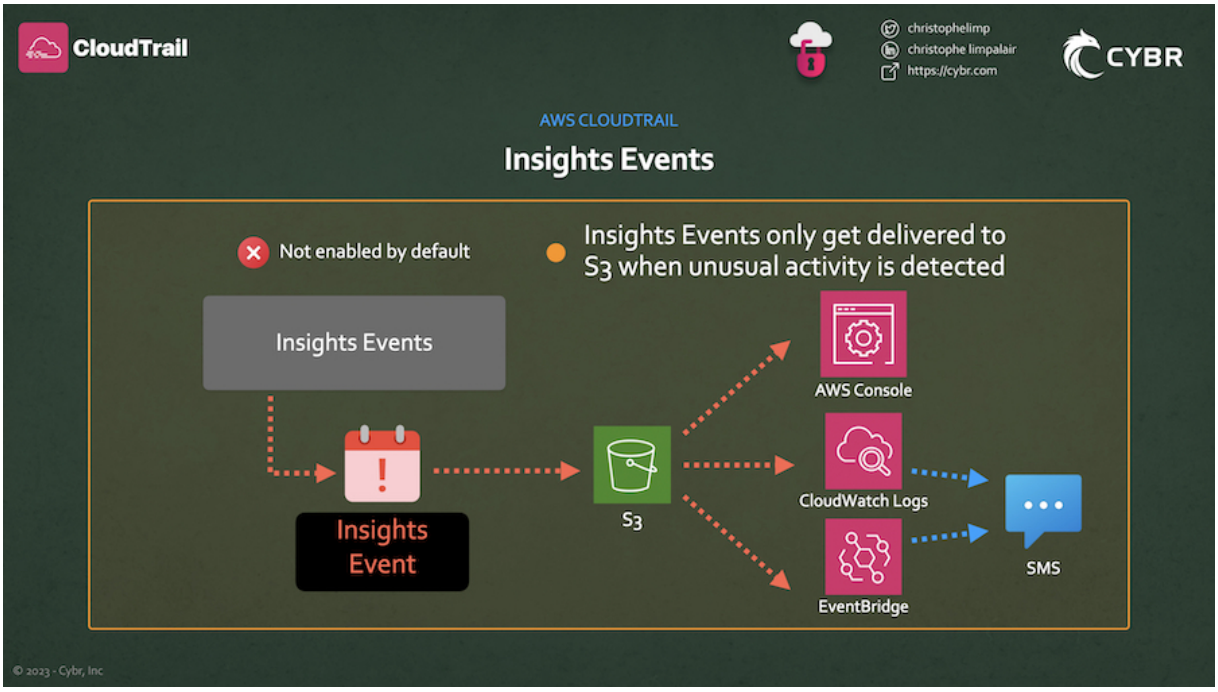
- The associated API
- Error code
- Incident time
- Statistics



CloudTrail Insights are not enable by default and they are an extra charge, but I'll show you how to enable and use them in the next section.

Once they are enabled, and CloudTrail detects unusual activity, it will deliver those events to your destination S3 bucket. This is another key difference because Insights event logs will only get delivered when an unusual activity is detected.

You can also view the information from the CloudTrail console, and you can configure CloudTrail to send Insights events to CloudWatch Logs, or you can create a rule in Amazon EventBridge to deliver Insights events. Both of those approaches could be used to generate an email or SMS notification whenever an Insights event is generated so that you can investigate.



Conclusion

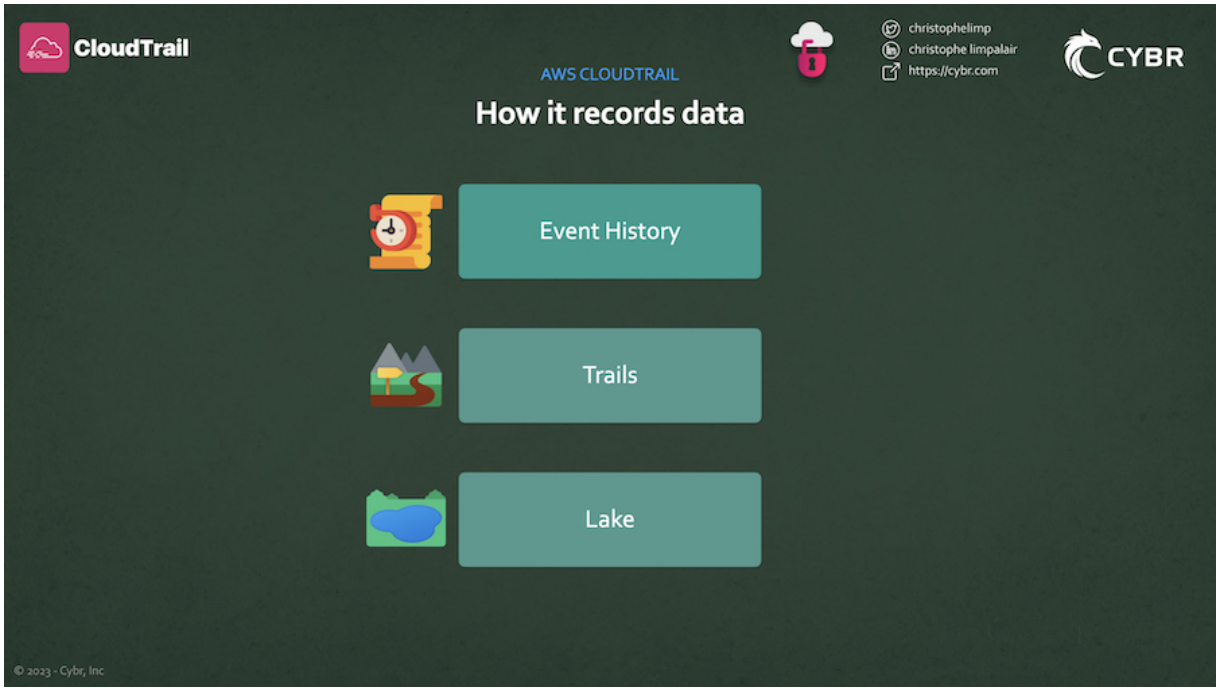
We've covered quite a bit of important information in this lesson, so feel free to either go back through it another time or use the sources I've included in the transcript to make sure that you have a solid grasp of what gets logged by Management Events versus Data Events versus Insights Events.

Then, go ahead and complete this lesson, and let's move on to the next!

The 3 ways of recording data

We've just learned about how CloudTrail can log Management, Data, and Insights Events. Now let's talk about the 3 ways it has to record this data.

The first way is Event History, the second is Trails, and the third is CloudTrail Lake.



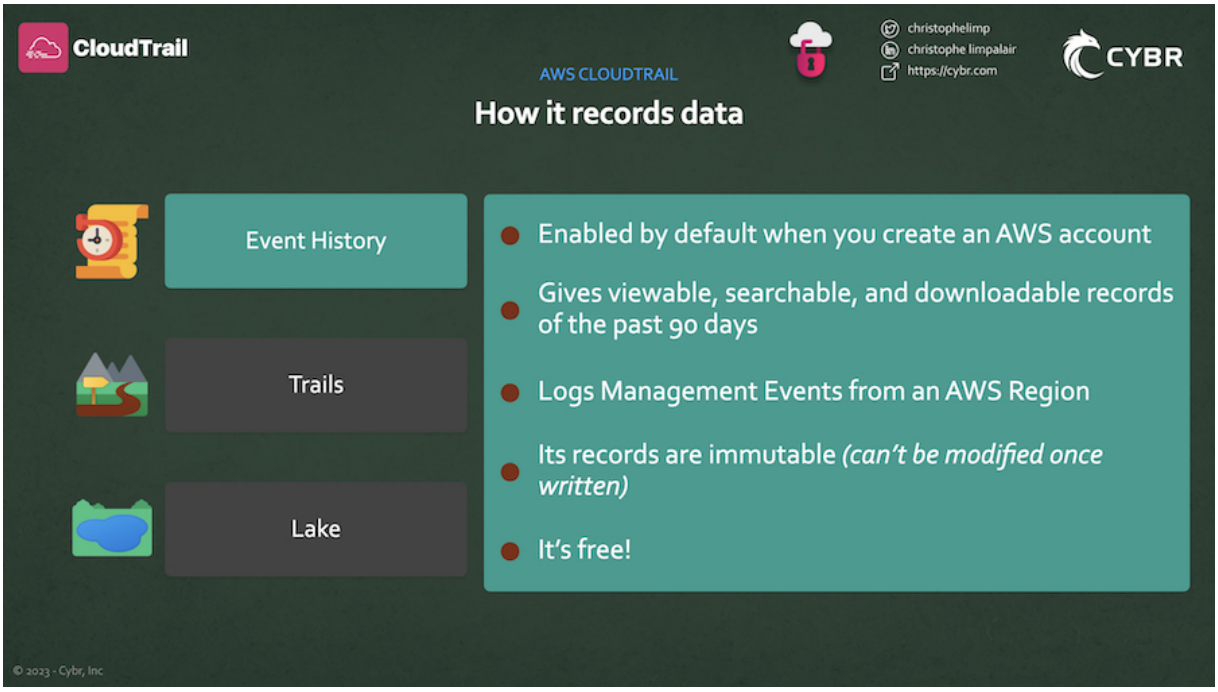
Event History

The Event History in CloudTrail is enabled by default when you create your AWS account. This gives you a viewable, searchable, and downloadable record of the past 90 days of management events in an AWS Region.

These records are also immutable, meaning that they can't be modified by anyone once they're generated. This is very important as it means you can can trust the data hasn't been tampered with after it was recorded.

The best part about Event History is that it's completely free.

In the next section of working with CloudTrail, we'll start by looking at Event History and how we can use it.



Trails

Next, and this is what most people know about CloudTrail because it's a core function of the service, we can record data by creating trails…

After all, this is how it gets the service name!

We'll take a look at how to create our first trail very soon in the next section, but it's important to understand that trails are what capture records of AWS activities that you can then store in Amazon S3, instead of just in Event History, or just for Management Events.

We talked about how Management Events are stored for 90 days in Event History, and you can create a trail to store those Management Events in a more permanent S3 bucket instead so that you don't lose valuable data after that 90 days.

Data Events and Insight Events can also be logged by trails if you configure that. So if you are interested in more than just Management Events, creating a trail would be a great way of doing that.

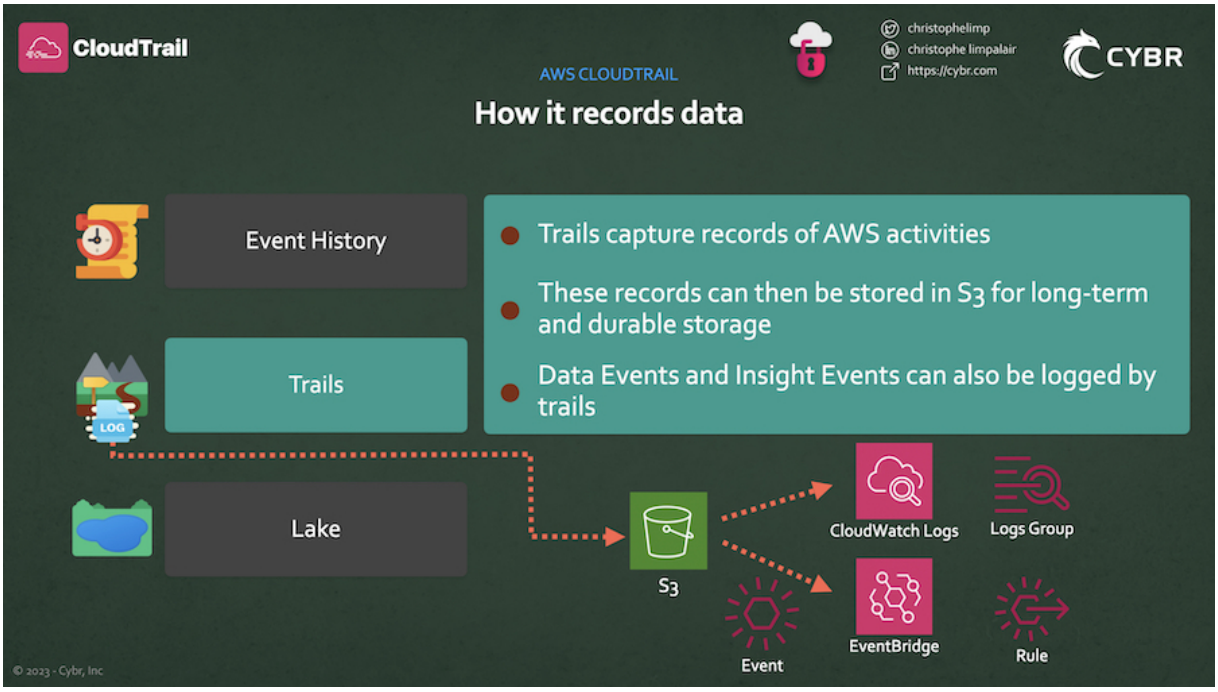
You create an Amazon S3 bucket, and you specify that bucket when you create a new trail, and the service will automatically deliver logs to that bucket.

On top of that, you can optionally send logs to CloudWatch Logs by creating a Log Group, and you can create Amazon EventBridge rules that use information from CloudTrail.

We'll talk more about why all of this is important later on with examples.

While you don't get charged by AWS for storing one copy of ongoing management events to an S3 bucket, you do get charged the S3 storage charges which are quite minimal, but keep that in mind.

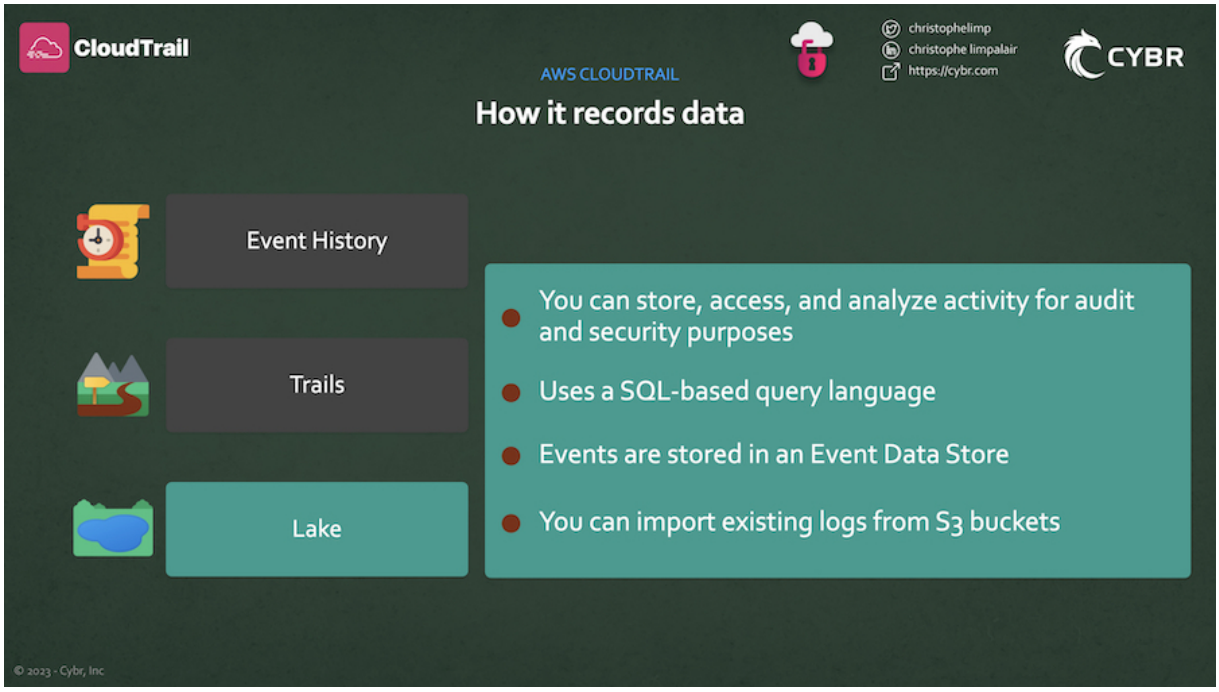
You also get charged for Data Events that get delivered to S3 from a trail.



CloudTrail Lake

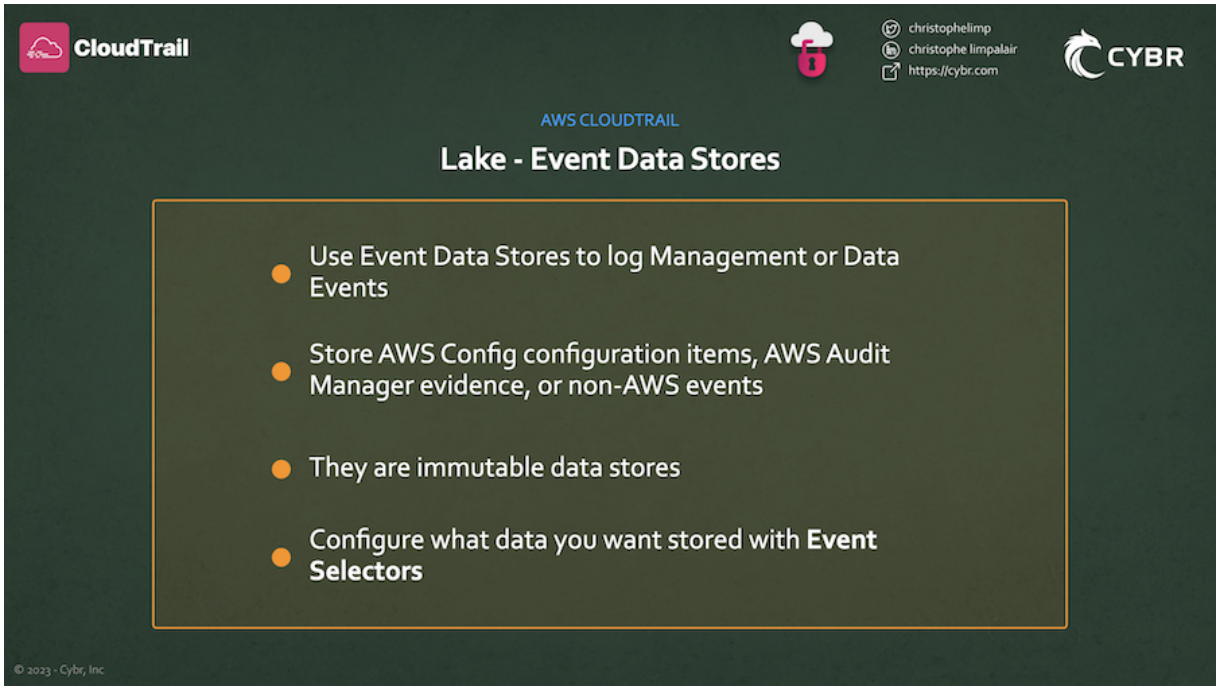
The third way of recording CloudTrail data is called CloudTrail Lake. Lake is a feature that lets you store, access, and analyze activity for audit and security purposes using a SQL-based query language.

With Lake, the events get stored into what's called *event data store*, and you can even import existing CloudTrail logs from your S3 buckets into an event data store.



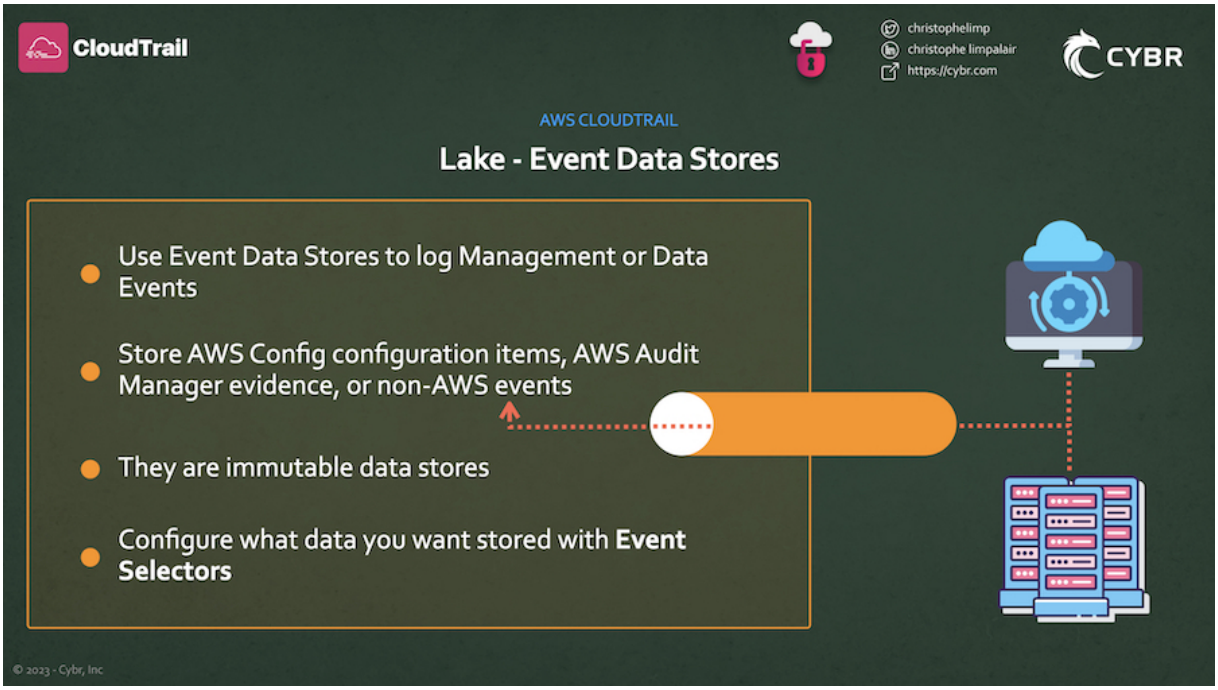
Event Data Stores

You can use Event Data Stores to log CloudTrail Management and Data Events, and you can also store AWS Config configuration items, AWS Audit Manager evidence, or other non-AWS events from 3rd party integrations.



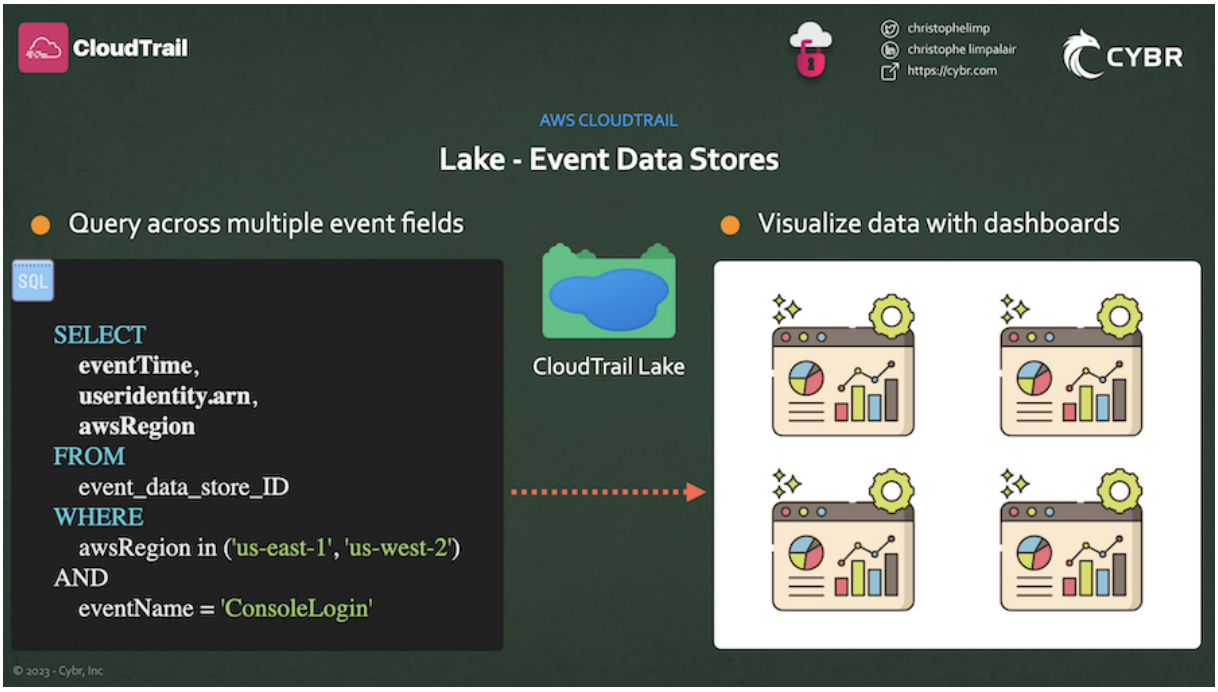
These Event Data Stores are immutable collections of event data, and you can choose exactly what data you want stored by configuring what's called `event selectors`.

An added benefit of using Lake is that it provides `integrations` which you can use to log and store activity data from outside of AWS. It could be a source from your on-prem VMs and containers, or even SaaS applications.



Once in Lake, you can query the event data using advanced SQL queries across multiple event fields, which is a lot more powerful than what you can do with Event History.

You can also visualize CloudTrail data by creating Lake dashboards, where each dashboard can be made up of multiple widgets that represent a SQL query.



While Lake is pretty awesome, my main gripe with it is that it can become very expensive if you have a lot of data. Check out the pricing page for more details, but at a high level, you get charged for data ingestion and storage, as well as data analysis.

CloudTrail is not real-time logging

As we wrap up this lesson, I saved an important bit of information for the end: keep in mind that CloudTrail is not considered to be real-time logging, because AWS says that:

> CloudTrail typically delivers events within an average of about 5 minutes of an API call. This time is not guaranteed.
> Source: https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-lake.html

So yes, there can be a bit of a delay from the moment you or an attacker makes an API call to the moment that you are even able to see it, and/or receive an alert. If you need to know more information about this you can check out the Service Level Agreement.

Conclusion

Alright, now that we've covered the 3 ways that CloudTrail records data, let's complete this lesson and let's move on to the next!

[Cheat Sheet] CloudTrail Events and Data Recording

Image link: https://cybr.com/wp-content/uploads/2023/11/aws-cloudtrail-1716x2048.jpg

Working with CloudTrail

Working with Event History

As we've talked about, part of CloudTrail is enabled by default when you create an AWS account, and that part includes enabling Event History.

Event History gives us 90 days of storage for collected Management Events, and it's a feature we can use to view, search, and download these records.

Event History in the Console

Let's take a look at how we can do that in the AWS console.

First, navigate to AWS CloudTrail.

From there, click on `Event history` in the left menu.

From there, you can search events in Event History by filtering for events on a single attribute. Let's take a look at an example:

Let's say that you want to see who recently made changes to a specific security group.

For the Filter, select `Resource name`. Then, for the search value, you'll want to input the security group ID: `ID_example_here`.

You can then enter a time range if you want, and if you do, click on `Apply`.

You can then click on the individual events to pull up more details about the event. If you click on `View event record`, you will see the exact JSON record with even more information.

It will show who made the change (the `principalId` and `ARN`, with which access key `accessKeyId`, when with `eventTime`, and what the changes were with `requestParameters`.

Closing out of that, if you select multiple events, they will show up side by side in the table so that you can compare values.

Click on the `Event name` hyperlink will take you to a page view dedicated to that event.

If we go back to the main `Event history` view, we can `Download events` as `CSV` or as `JSON`, and we can also `Create Athena table`, which we won't demonstrate in this lesson.

The last thing I'll mention before we move on is that you can customize this table and what you see by clicking on the cog icon next to the pagination numbers, where you can enable or disable the visible columns, you can change how many events are shown per page, and more.

AWS CloudTrail Lookup-Events

Of course, AWS CloudTrail exposes a public API, so we don't just have to use it through the AWS console. We can also query it through the AWS CLI using the AWS CloudTrail Lookup-Events.

Let's take a look at some examples of how we can use this.

Make sure you have the AWS CLI installed. I recommend using the AWS CLI version 2, but version 1 will also work.

If you don't already have the AWS CLI installed, which you can check by opening up your terminal and typing in:

```
aws help
```

If that opens up the help menu, then you have the CLI installed and you can proceed. (Type `q` to exit out of the menu)

If you get an error and don't see this, please check out this page for instructions on how to install the CLI, then come back here.

Now, to see a list of the latest event, you can type in the command:

```
aws cloudtrail lookup-events --max-items 1 --output json
```

Results:


```
{
    "Events": [
        {
            "EventId": "b2718728-e044-416a-967c-d7d7a2fd9499",
            "EventName": "GetBucketAcl",
            "ReadOnly": "true",
            "EventTime": "2023-11-10T14:22:38-07:00",
            "EventSource": "s3.amazonaws.com",
            "Resources": [],
            "CloudTrailEvent": "{\\"eventVersion\\":\\"1.09\\",\\"userIdentity\\":{\\"type\\":\\"AWSServ
ice\\",\\"invokedBy\\":\\"cloudtrail.amazonaws.com\\"},\\"eventTime\\":\\"2023-11-10T21:22:38Z\\",\\"eve
ntSource\\":\\"s3.amazonaws.com\\",\\"eventName\\":\\"GetBucketAcl\\",\\"awsRegion\\":\\"us-east-1
\\",\\"sourceIPAddress\\":\\"cloudtrail.amazonaws.com\\",\\"userAgent\\":\\"cloudtrail.amazonaws.com
\\",\\"requestParameters\\":{\\"bucketName\\":\\"aws-cloudtrail-logs-272281913033-419e1363\\",\\"Host
\\":\\"aws-cloudtrail-logs-272281913033-419e1363.s3.us-east-1.amazonaws.com\\",\\"acl\\":\\"\\"},\\"resp
onseElements\\":null,\\"additionalEventData\\":{\\"SignatureVersion\\":\\"SigV4\\",\\"CipherSuite
\\":\\"ECDHE-RSA-AES128-GCM-SHA256\\",\\"bytesTransferredIn\\":0,\\"AuthenticationMethod\\":\\"AuthHeade
r\\",\\"x-amz-id-2\\":\\"redacted\\",\\"bytesTransferredOut\\":558},\\"requestID\\":\\"ZQRKredacted
\\",\\"eventID\\":\\"b2718728-redacted\\",\\"readOnly\\":true,\\"resources\\":[{\\"accountId\\":\\"27228
1913033\\",\\"type\\":\\"AWS::S3::Bucket\\",\\"ARN\\":\\"arn:aws:s3:::aws-cloudtrail-logs-272281913033-4
19e1363\\"}],\\"eventType\\":\\"AwsApiCall\\",\\"managementEvent\\":true,\\"recipientAccountId\\":\\"272
281913033\\",\\"sharedEventID\\":\\"eb511ecc-redacted\\",\\"eventCategory\\":\\"Management\\"}"
        }
    ],
```

```
    "NextToken": "eyJOZXh0VG9rZW4iOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAxfQ=="
}
```

We get back data like the `EventId`, `EventName`, `EventTime`, `EventSource`, and a `CloudTrailEvent` full of information.

- For a list of output fields and what they mean, check out this documentation: https://docs.aws.amazon.com/awscloudtrail/latest/userguide/view-cloudtrail-events-cli.html#view-cloudtrail-events-cli-output-fields*

Instead of specifying a max limit for items returned, we can specify a time range, like this:

```
aws cloudtrail lookup-events --start-time <timestamp> --end-time <timestamp>
```

The timestamps take in UNIX time values, and you can specify the date, month, and year values separated by forward slashes or hyphens.

```
1422317782
1422317782.0
01-27-2015
01-27-2015,01:16PM
"01-27-2015, 01:16 PM"
"01/27/2015, 13:16"
2015-01-27
"2015-01-27, 01:16 PM"
```

So for example, we can do:

```
aws cloudtrail lookup-events --start-time "2023-11-10, 10:00 AM" --end-time "2023-11-10, 10:30 AM" --pro
file test --output json
```

(Ctrl or Cmd + C to exit if there's a lot of information)

We can also query by using a single attribute, like this:

```
aws cloudtrail lookup-events --lookup-attributes AttributeKey=<attribute>,AttributeValue=<string>
```

Attributes are key/value pairs, so you would specify an `AttributeKey` and then the `AttributeValue`.

For the `AttributeKey`, you can use these values (and they are case sensitive):

- AccessKeyId
- EventId
- EventName

- EventSource
- ReadOnly
- ResourceName
- ResourceType
- Username

As a practical use case, let's say that you think an access key has potentially been compromised and has been used by a threat actor. You can query the last 10 events related to that access key like this:

```
aws cloudtrail lookup-events --lookup-attributes AttributeKey=AccessKeyId,AttributeValue=EXAMPLE_KEY_ID_
HERE --max-items 2
```

Or, if you believe that a role has been used for malicious purposes, you can search for Events related to that role, like:

```
aws cloudtrail lookup-events --lookup-attributes AttributeKey=ResourceName,AttributeValue=CloudTrail_Clo
udWatchLogs_Role --max-items 2
```

As another similar example, you can search for events related to specific users by using the `Username` `AttributeKey`:

```
aws cloudtrail lookup-events --lookup-attributes AttributeKey=Username,AttributeValue=christophe --max-i
tems 2
```

If one of these commands returns multiple page's worth of results, you can use the `next-token` and pass that into a `--next-token` parameter, like this:

```
aws cloudtrail lookup-events --lookup-attributes AttributeKey=Username,AttributeValue=christophe --next-
token=kbOt5LlZe++mErCebpy2TgaMgmDvF1kYGFcH64JSjIbZFjsuvrSqg66b5YGssKutDYIyII4lrP4IDbeQdiObkp9YAlju3oXd12
juy3CIZW8=
```

LookupEvents API operations

Apart from using the CLI, we can also make API calls via scripts, apps, or third-party software. Like if we're using the AWS SDK for .NET, Java, JavaScript, Python, etc…, we can use this API.

It's very similar to what we just saw talking about the CLI because they use the same API, but if you want more information, check out this documentation page.

Important limitations of Event History

There are a few important facts to keep in mind about Event History:

1. When using the AWS Console, you can only view Management Events, not Data Events
2. It only stores data for up to 90 days and then you lose that data
3. You can only use it to search data from a single account, and it will only return events from a single AWS region
4. You can't query multiple different attributes, so unlike CloudTrail Lake, search-ability is very limited
5. You can't exclude AWS KMS or Amazon RDS Data API events

Conclusion

Overall, Event History can give you a good starting point and it can be useful when you need to run simple queries against your data. It can give you a good starting point for security or troubleshooting investigations, but you will likely get to a point where you find it too limiting, and that's when you could turn to something like CloudTrail Lake, CloudWatch Logs Insights, or Athena.

Also, considering some of Event History's limitations, especially the maximum of 90 days of storage and the lack of customization, you will want to consider creating one or more trails to overcome those limitations.

With that, let's complete this lesson so that we can move on and learn more!

Create your first trail

Now that we've seen Event History being used in action, and we understand its limitations, we need to take a look at how we can create trails.

Let's pull up CloudTrail in the AWS console.

Click on `Create a trail`.

Give this trail a name, like `cybr-sample-trail`.

Then create the trail.

Click on your newly created trail, and let's `Edit` the `General Details`.

You'll notice that I have a greyed-out option:

`Enable for all accounts in my organization`

This is a useful feature if you are using AWS Organizations, but in this case I'm actually using a child account that's part of my parent organization, which is why this is greyed out.

But in any case, the quick start approach to creating a new trail automatically creates a new bucket in S3 with a unique name. You can always change this to one of your existing buckets if you already

have one.

We can also enable log file encryption using SSE-KMS, and we know what that means because we learned about it earlier in the course.

Again, the quick start approach disables this by default, but otherwise, this feature is actually enabled by default and if we were to enable it now, it would ask us to either create a new key, or use an existing one. This is recommended for production, but I'll leave it disabled for my presentation.

We also have a checkbox to enable `Log file validation`. Again, you would normally want to enable this for production reasons, but I'll leave it off for this example.

Next we have the ability to enable `SNS notification delivery` which would notify you each time a log is delivered to your bucket.

This is not very practical for most people and would result in a bunch of notifications, and so instead it's generally better to enable notifications only for important events.

If you make modifications then go ahead and `Save changes`, but otherwise, you can cancel out.

Congratulations, you've now successfully created your very first CloudTrail log!

From this dashboard, you can see general information about this trail, including that this is a multi-region trail, meaning that it will log events across all Regions in in your AWS account.

> As a quick aside, AWS' documentation says: To log events across all Regions in all AWS partitions in your account, create a multi-Region trail in each partition. If you're not familiar with partitions, AWS currently offers 3 partitions: 1) an aws partition for AWS regions, 2) an aws-cn partition for China, and 3) an aws-us-gov partition for AWS GovCloud US regions. So unless you're in China or using a GovCloud AWS account, this doesn't apply to you. All you need to know is that you either create individual trails per region, or you enable multi-region trails to automatically do that for you

Logs in Amazon S3

Before we wrap up, let's head over to Amazon S3 so that we can see the Bucket that was created for us.

If we click on that bucket, we should already see some AWS logs. Open up those directories until you see a few different objects.

As you can see, we already have some log files. The problem is that these are compressed files, which means we would have to download them, extract them, and then sift through a bunch of hard-to-read information.

That’s why you would want to use something else like CloudWatch Logs, CloudTrail Lake, Athena, or even a third-party tool, in order to query this data.

Conclusion

We’ll take a look at some of that in the following lessons. For now, complete this lesson and I’ll see you in the next.

Working with CloudTrail trails

We looked at how we can create a trail using the AWS console, and we set up that trail to store files in Amazon S3.

In this lesson, let’s talk about:

- How to locate log files
- How to download files
- How to search through those files

Locate log files

In the prior lesson, we saw where our log files were stored, but we only had a few files created. What about when you’ve had CloudTrail enabled for months? How do you know which files to look at?

Log file storage follows this path:

```
bucket_name/prefix_name/AWSLogs/Account ID/CloudTrail/region/YYYY/MM/DD/file_name.json.gz
```

As an example:

```
aws-cloudtrail-logs-272281913033-419e1363
AWSLogs/272281913033/CloudTrail/us-east-1/2023/11/10/10/272281913033_CloudTrail_us-east-1_20231110T2225Z
_LMThgKKp783oekGe.json.gz
```

So you can rely on this format remaining constant, which means you can use this when you are manually looking for files, or if you’re writing any sort of software that will ingest these files.

Download log files

To download these files, you have a few options:

1. Use a JSON viewer add-on to open the files in your browser

2. Download the files from S3 – you can do that by selecting them in the console then clicking on `Download`, or by using the CLI or API

There are also many 3rd party vendors that let you upload your log files to then be able to process them, and you can see those on this page.

Search through log files

Instead of demonstrating how to use one of those 3rd party solutions, let's see how we can use our local terminal.

To get started, make sure you download one of your log files from S3.

Then, open your terminal and install `jq` if you don't already have it. There are multiple ways of installing it with instructions on this page.

For example, on MacOS, you could use Homebrew:

```
brew install jq
```

I already have it installed which I can tell by typing:

```
which jq

/opt/homebrew/bin/jq
```

`jq` is a lightweight command line tool that we can use to parse, filter, and manipulate JSON data which is what these log files are made up of.

Navigate to where you've downloaded your log file in the terminal.

```
cd ~/Downloads
```

Then type:

```
cat file-name.json
```

```
cat 272281913033_CloudTrail_us-east-1_20231110T2220Z_qHVjJUfkhQkCyZks.json
```

If your files did not download extracted and have an extension of `.json.gz`, you'll first need to extract the files which you can do either with something like 7-Zip or using `gunzip`. It really depends on what

operating system you're using, but since you've probably extracted archives before, I won't show how. Leave a comment if you need help.

As you can see, catting this file is a mess. It's formatted in JSON, but there's no whitespace at all, so it's very difficult to read.

Since it's JSON and we have `jq` installed, let's use it!

To make running commands easier, let's make a directory:

```
mkdir cloudtrail
```

Then move our file into it:

```
mv 272281913033_CloudTrail_us-east-1_20231110T2220Z_qHVjJUfkhQkCyZks.json cloudtrail/
```

Then let's go into that directory:

```
cd cloudtrail
```

We're now ready to search this file. Let's start by searching for any and all user identities in this file:

```
find . -type f -exec jq '.Records[].userIdentity.arn' {} \\;

"arn:aws:sts::272281913033:assumed-role/lab/christophe"
```

This will search all files in our directory, then run the `jq` command looking for `userIdentity.arn`

Please note that this is case sensitive, so if you capitalize the `User` for example, it won't find anything.

We see our `christophe` identity returned and the complete ARN for who are what made the call, which in this case is an assumed role.

That's a helpful starting point, but what other valuable information can we extract from this?

Let's run this command:

```
find . -type f -exec jq -r '.Records[] | [.eventTime, .sourceIPAddress, .userIdentity.arn, .eventName] |
join(" -- ")' {} \\; | sort
```

This command will return:

```
2023-11-10T22:14:31Z -- cloudtrail.amazonaws.com --  -- GetBucketAcl
2023-11-10T22:18:17Z -- health.amazonaws.com -- arn:aws:sts::272281913033:assumed-role/lab/christophe --
```

```
DescribeEventAggregates
```

…which contains the:

- Event Time
- Source IP address (or in this case the domain for the service)
- User identity ARN which tells us what type of identity it is
- Event name

Conclusion

As you can see, this is quite a different approach to using the Event History API, and it already gives us more flexibility, *but*, it's also not as user-friendly and requires terminal experience.

If you're not used to working from the terminal, writing scripts, or using something like `jq`, this approach has a learning curve.

Even if you are used to it, as you'll see once we get to the CloudTrail Lake lessons, there are many advantages to using that service instead of doing this.

With that said, this option is available and is a valid way of analyzing your logs.

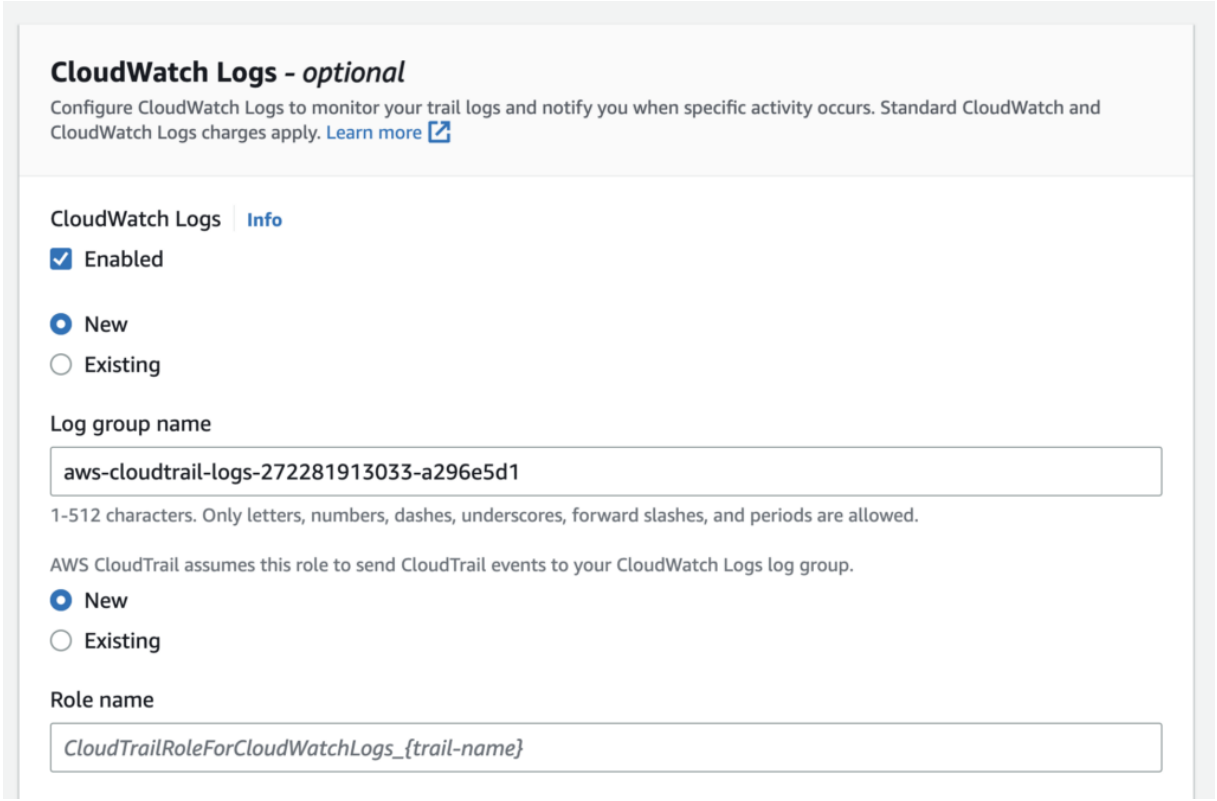
With that, let's complete this lesson and let's move on!

Working with CloudWatch Logs and SNS notifications

While we can enable pushing logs from CloudTrail to CloudWatch as we're initially setting up our trail, we can also do it retroactively. If you're not already on this page, go to your CloudTrail dashboard, and then click on your existing trail that we created in a prior lesson.

Enable CloudWatch Logs

Below `General details`, click on `Edit` under `CloudWatch Logs`.

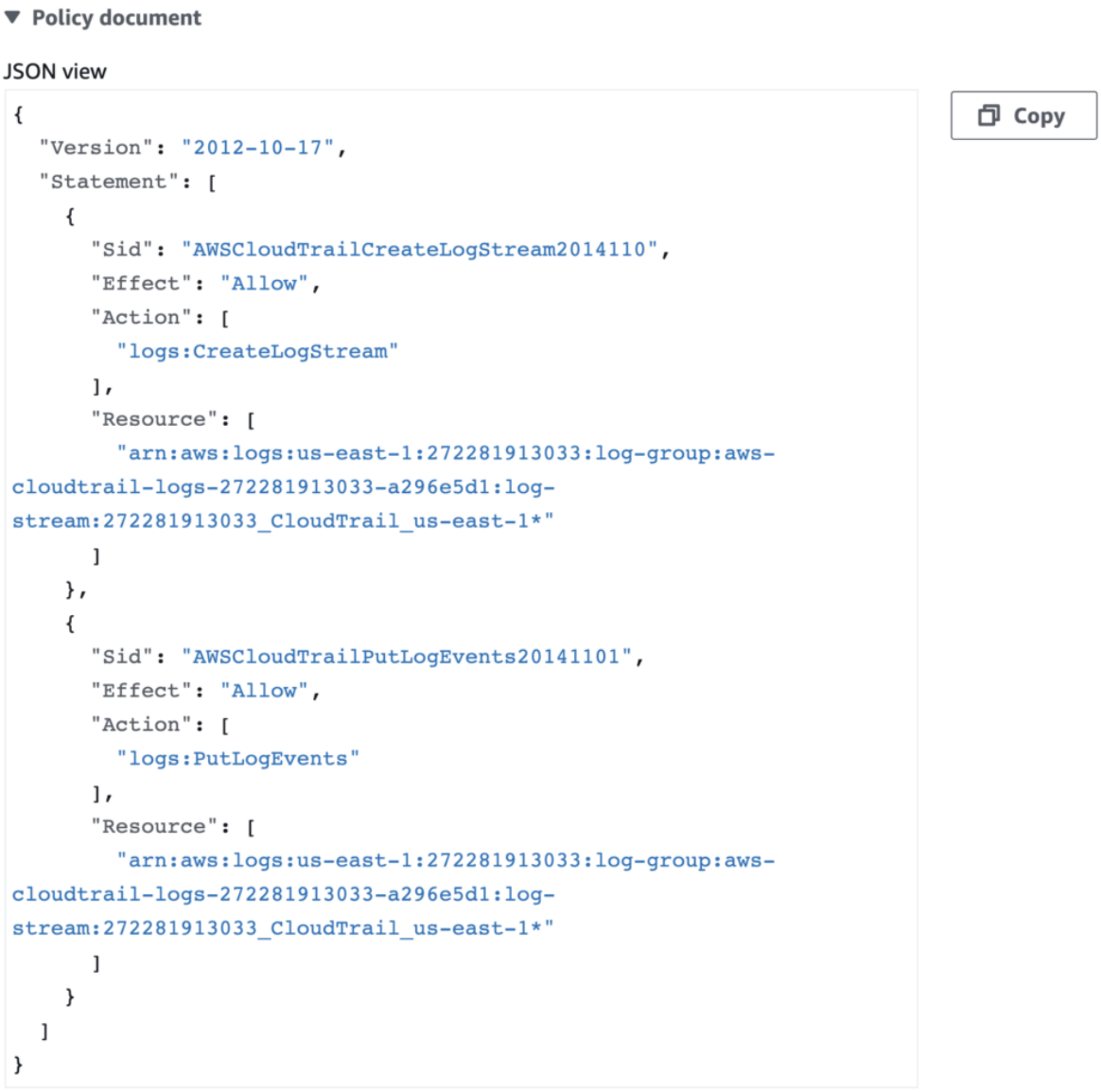


This is how we can push logs to CloudWatch Logs groups so that CloudWatch will monitor our trail logs and notify us when a specific event happens.

If you plan on using this, you may want to check CloudWatch pricing first.

Then, check the `Enabled` box. Assuming that you don't already have a CloudWatch log group, you'll want to give it a unique name or use the auto-generated name.

This requires that CloudTrail assume a role with proper permissions to be able to push events to the group, so we can create that new role on this page by giving it whatever name we want (like `CloudTrailRoleForCloudWatchLogs`), and we can expand the policy document to view what gets created by default.



Click on `Save changes` and give it a moment to create these new resources.

After a brief moment, we'll see our new CloudWatch Logs information, and we're all set up!

CloudWatch Logs

In the CloudWatch Dashboard, head over to Log Groups in the menu. You should see your `aws-cloudtrail-logs-...` log group. If we click on it, we should see `Log streams` that we can click on.

Inside of that, we'll find the `Log events` for that `Log group`, and we can expand each entry for more information.

This information is much more easily digestable, and more importantly, it's searchable!

For example, I could search for `christophe`, and it will filter to only show events that have to do with my user.

Not only is this super helpful for incident response and investigation, but it also means that we can set up filters for specific keywords or actions that trigger notifications to be sent. Let's take a look.

Amazon SNS

Let's head on over to Amazon SNS, or Simple Notification Service.

The first step is to create a new topic. You can name this whatever you'd like (ie: `CybrTopic`).

Feel free to look through the other settings but for this we'll leave everything to the defaults.

When you're ready, `Create Topic`.

We then need to `Create subscription`.

You have access to multiple different protocols. For this, I just want to send emails to my email address, so I'll select `Email` and I'll set the `Endpoint` to my email address.

Keep in mind that you *must* confirm your subscription from your inbox. This is to prevent spam.

Go ahead and `Create subscription` and then check your email inbox to confirm.

After you confirm, which I've done off-screen, go back to CloudWatch.

CloudWatch Metrics

Back in CloudWatch, make sure you're looking at your `Log groups`. Select the CloudTrail log group, and click on the `Actions` dropdown menu, then click on `Create metric filter`.

This is where we can decide what to filter for. Oftentimes we'd want to filter for `ERROR` or `AccessDenied` or something similar to that. For our example, I'll just filter for `christophe`.

Before committing to this, you can test your pattern by selecting log data to test and clicking on `Test pattern`.

We can see a bunch of log event messages, and all of these should contain my username.

Click on `Next` when you're ready.

Set a filter name, like `christophe-filter`.

For `Metric details`, I'll just use:

- `Metric namspace` `christophe`
- `Metric name` `christophe`
- `Metric value` `1` → which represents the value published to the metric name when there's a positive filter pattern match

Then click on `Next`, review your settings, and `Create metric filter`.

We now have a metric name, which means we can move on to create an `Alarm`.

CloudWatch Alarms

Click on `All alarms` in the menu, and then `Create alarm`.

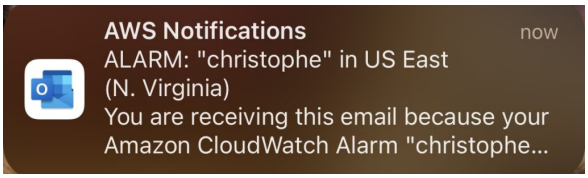
Select the metric you created — but keep in mind it can take a minute or so for it to show up. If you don't see it and can't find it by searching, then refresh until you do.

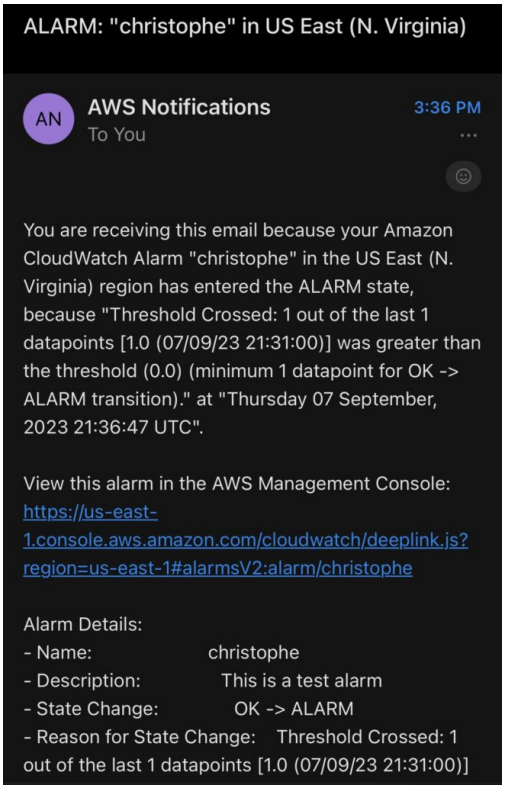
You can then configure the conditions for the alarm based on a few different factors. For example, you could trigger an alarm whenever there's an `Anomaly detection`, or whenever it exceeds or is lower than a certain threshold, etc… I'll just set mine `Greater` than `0` and click on next.

Make sure your `Alarm state trigger` is set to `In alarm`, and then `Select an existing SNS topic` searching for the topic you created earlier. Click on `Next`, name your alarm, and set a description if you'd like (ie: `This is a test alarm`).

Click `Next`, review, and then `Create alarm`.

Wait for 5 minutes (unless you changed your condition), and you should receive an email alarm.





Conclusion

I'm now going to quickly turn off this alarm so I don't get flooded with notifications, but congratulations! You've created an automated alarm to go through whenever a certain condition is met through CloudTrail logs and by using CloudWatch log groups as well as SNS.

As you can imagine, setting up effective alarms and monitoring for AWS accounts can take a long time. We're already a few minutes in for just an example — so it can take days, weeks, or even months to develop and architect effective monitoring, logging, and alerting. But this lesson gives you a starting point for implementing this in your own environments, so be sure to apply what you just learned today!

Also, remember that CloudTrail is going to continue creating logs and putting them in S3, and then pushing to CloudWatch, so if you want to stop getting charged, be sure to delete everything we created, or stop the trail from producing.

After that, go ahead and complete this lesson, and I'll see you in the next!

Working with CloudTrail Insights

We talked about how CloudTrail has a feature called Insights that can help us make sense of logged data.

As a quick reminder, Insights uses machine learning to automatically analyze write management events from CloudTrail Trails, and it looks for anomalies. It does that by creating baselines of normal, expected behavior, and whenever there's a significant change from that baseline, it generates an Insight Event.

Let's see how we can enable and use this functionality in the AWS Console.

From the CloudTrail dashboard, you can enable Insights either as you're creating a trail, or you can go back and edit existing trails to enable this feature.

Since we already have a trail created, let's edit it.

Enable Insights on a Trail

Select your trail, and scroll down until you see a section for `Insights events`.

Click on `Edit` .

Select `Insights Events` which will display more options.

This is where you can decide whether you want to enable this for `API call rate`, `API error rate`, or both.

Because AWS charges for this per 100,000 events analyzed, *per* Insight type, this can affect the cost so that's a decision you have to make. For this demo, I'll enable both and then click on `Save changes`.

That's it! You've enabled Insights for this specific trail. If you have other trails, you could go in and enable it for those trails as well.

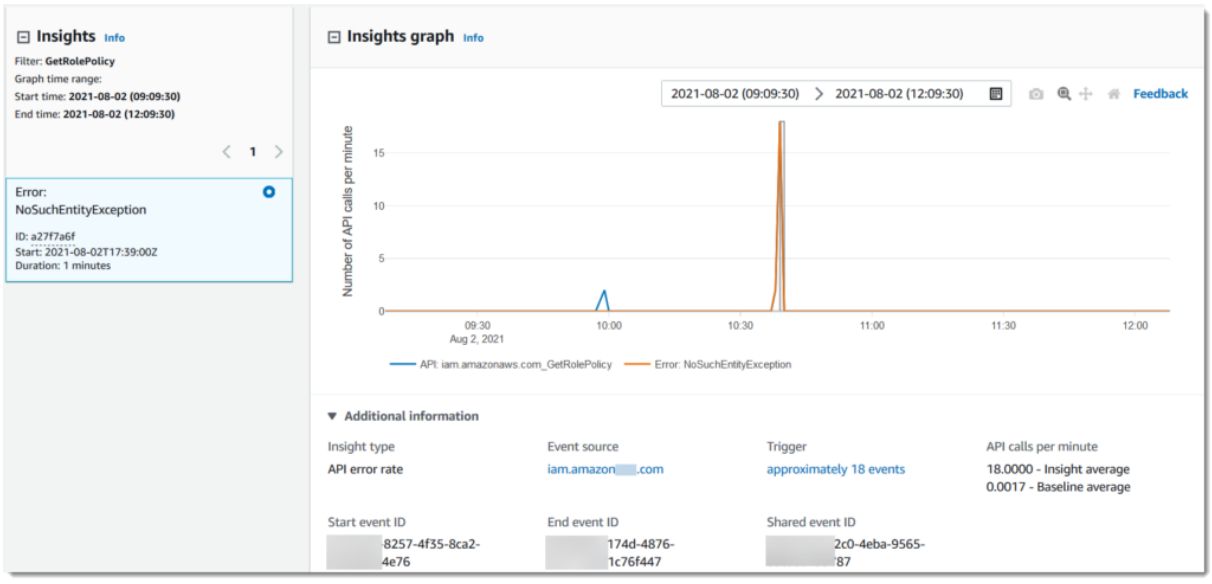
Once you enable it, a few things happen behind the scenes:

1. AWS will automatically start crunching data in the background
2. If or when CloudTrail detects an anomaly, it will generate an Insight Event and send it to:
    a. The S3 bucket associated with that trail – and it will create a separate folder named `CloudTrail-Insight`
    b. EventBridge (which used to be called CloudWatch Events)
    c. If you've enabled it, to the associated CloudWatch Logs Group
3. CloudTrail can start detecting and showing anomalies in the console within 30 minutes, but it could take up to 36 hours when you enable it for the first time

You'll be able to see them in the Insights dashboard by clicking on `Insights` in the left menu, or also by going back to the main CloudTrail dashboard.

Instead of waiting 36 hours to produce this video, let me show you some screenshots of what Insights Events could look like:

API error rate example



Source: https://docs.aws.amazon.com/awscloudtrail/latest/userguide/logging-insights-events-with-cloudtrail.html

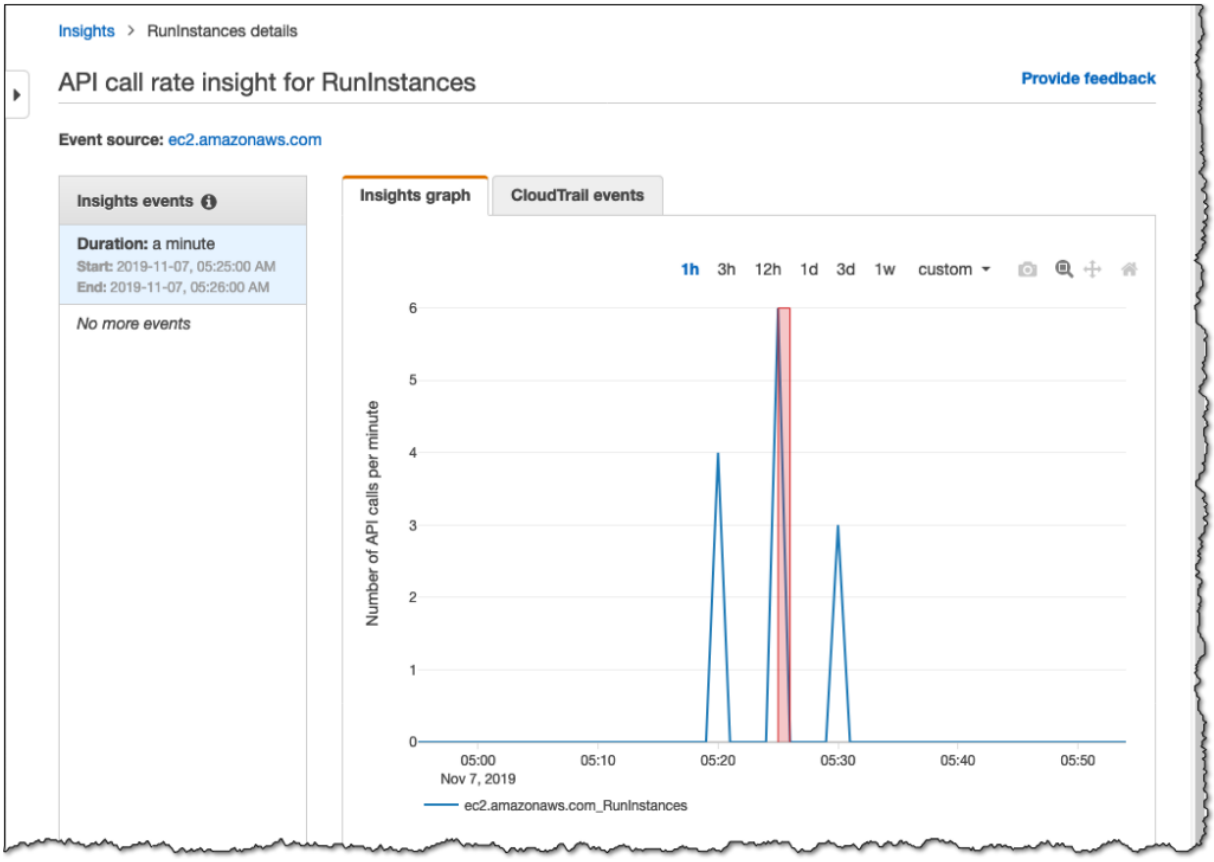
Here we have an Insight Event generated for an `API error rate`.

Someone or something is attempting to use the `GetRolePolicy` API call, which is an IAM API call an attacker could try to use to get information about the IAM policy associated with a role.

However, that API call is resulting in a `NoSuchEntityException` error, and we can see from the graph that normally we get 0 to maybe 2 such responses for hours at a time. But then, all of a sudden, we see 18 of these errors in less than a minute.

That tells us something is going on, and it could even be just an operational issue, it may not be a security issue, but this gives us practical information that we can then use to investigate.

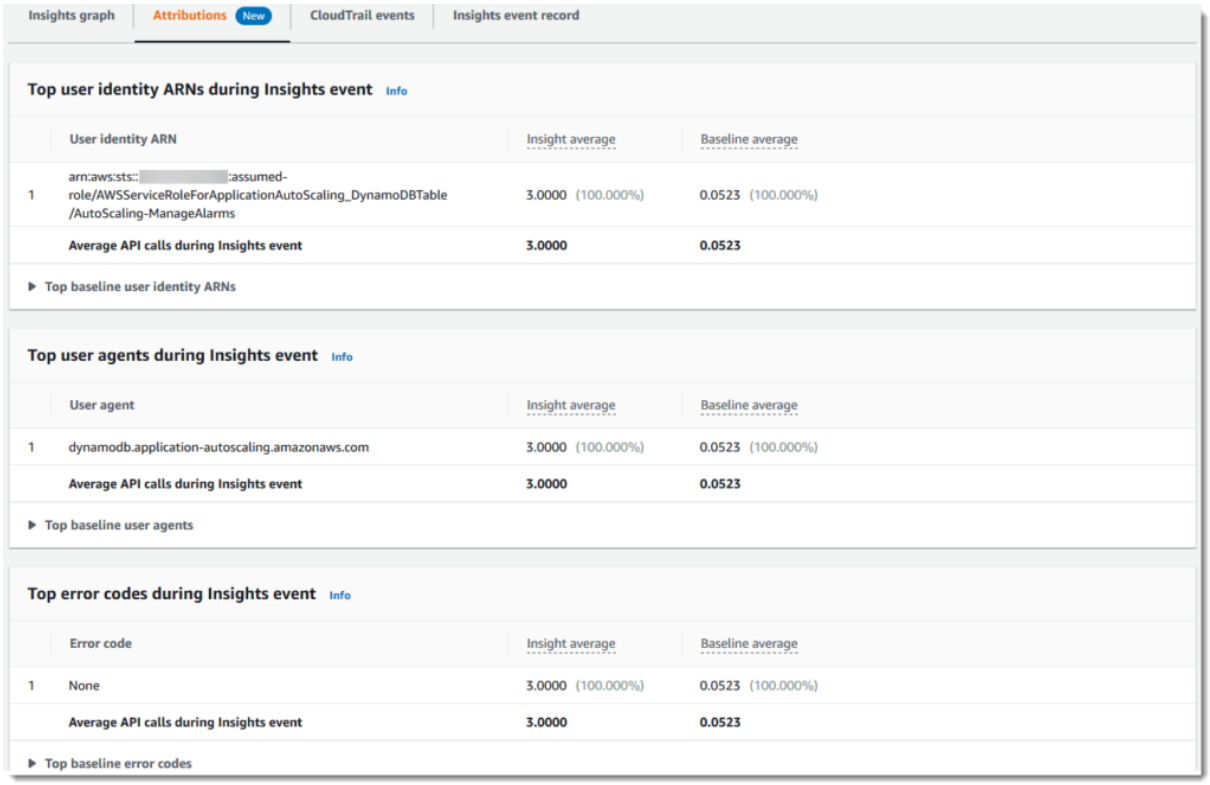
API call rate example



Source: https://aws.amazon.com/about-aws/whats-new/2019/11/aws-cloudtrail-announces-cloudtrail-insights/

Here's another example for a different API call of `RunInstances`. Instead of this being captured from an `API error rate`, this is captured by `API call rate`. So if we had only enabled `API error rate` when configuring our Insights settings earlier, then we would not have seen this type of event being generated.

Attributions



Source: https://docs.aws.amazon.com/awscloudtrail/latest/userguide/logging-insights-events-with-cloudtrail.html

Beyond graphs, we can also view `Attributions` details as we see them in this screenshot.

This page gives us the `Top user identity ARNs during an event`, we also get the `Top user agents` and `Top error codes`, etc…

Of course, this is a screenshot from the AWS Console, but we can also query Insights Events from other sources, including the `LookupEvents` API, just like we saw a few lessons ago when we were querying and filtering through the AWS CLI.

Conclusion

As you can see, Insights is easy to implement, and it can provide quite a bit of value. The main decision you then have to make is whether it's worth the cost for you.

But that's it! With that, go ahead and complete this lesson, and I'll see you in the next!

Working with CloudTrail Lake

We've looked at a few ways of querying our data from CloudTrail, but we've seen limitations with each approach. In this lesson, let's get started with CloudTrail Lake.

AWS recommends a 3-step process to get started:

1. Create an event data store
2. Run SQL queries
3. Optionally, add an integration

You'll remember that Lake collects and stores data in event data stores, which is what we can then run our queries against. So let's start by clicking on `Create event data store`.

Configure event data store

For the `Event data store name`, you can use whatever you'd like. I'll name mine `CybrDemo`.

We can then choose between pricing options of:

- One-year extendable retention pricing
- Seven-year retention pricing

I'll keep the selection to one-year.

Below that, you can either keep the default `1 year` `Retention Period` or you can extend it to 3 years, 10 years, or a custom period less than 10 years.

I'll keep the default of 1 year selected.

For `Encryption` we should use an AWS KMS key as a best practice, but by default, data is encrypted with an AWS-managed KMS key so I'll leave that disabled for our demo.

Next we can enable `Lake query federation` if we'd like, which lets you run SQL queries against your event data store by using Amazon Athena. I won't enable this for the demo, but feel free to do that if you'd like to play around with Athena.

I'll skip tagging and click on `Next`.

Choose events

On this next screen, we can select the `Event type(s)` that we want to add to our event data store. We can either do:

- AWS events, or
- Events from integrations

The second option is if we want to use this for apps that are outside of AWS, such as on-prem.

If we keep the selection to `AWS events`, then we get access to select between the type of AWS events we want to store:

- CloudTrail events
- CloudTrail Insights events

- Configuration items

The `Configuration items` option requires that we have AWS Config enabled since it works with AWS Config.

Let's select `CloudTrail events`, and then let's configure the events we want to capture between:

- Management events
- Data events
- Copy trail events
- Enable for all accounts in my organization (which I can't do since I'm using a member account)

By now we know the difference between management and data events, so I'll leave this selection up to you, but I'll select only `Management events`.

There's also a third option which is to copy events from an existing S3 bucket — so if you're enabling this after having collected some data from trails already, then you can import those events.

That's what I'll do. I'll select `Copy trail events` in addition to `Management events` which will instruct CloudTrail to grab a copy of the management data from our existing trail.

Because we're going to be ingesting existing data, I don't need or want to ingest additional events, and so I'll deselect the `Ingest events` option under `Additional settings` which will prevent this from collecting new data on top of what we're importing.

We then need to select a `trail event source` and I'll select my existing trail. It will pre-populate the S3 location, and then we can specify a time range of events.

I'll enter a `Custom range` of `1 month`.

I'll leave the default selection of `Create a new role` for `Choose IAM role` and I'll `Enter IAM role name` of `CybrDemo`.

You can expand the `Permission policies` to view what will get created, and then you can click on `Next`.

You'll get a warning about pricing since Lake can get expensive if you have a whole lot of data, especially because ingestion is metered on unzipped data which makes the data quite a bit larger than what you see on the S3 storage.

I'm fine with this so I'll check the box that says `Copy trail events to Lake` and then I'll `Copy events`.

Review your settings and then click on `Create event data store`.

On the next dashboard, you'll be able to see `General details`, what data is included, and the `Event copy status` where you can keep track of the data ingestion. You shouldn't have too much data to copy over, so this should finish within a few minutes.

Refresh until you see `Completed`.

Running queries

Now that we've created our first event data store, we're ready to run queries against that data.

Open up the left menu (if it collapsed) and click on `Query` under `Lake`.

From there, you'll be greeted with an `Editor` tab where you can select the `Event data store` from which you want to run your queries, `Event properties` that you'd like to use in your query, and then the `Output` of that query.

To save some time, click on the tab that says `Sample queries` so that we can use pre-made ones.

Sample query #1

Feel free to pause here and go through this list to see what all you can run, but I'll start off by searching for `Top APIs aggregated by source`.

Click on the query name and it will automatically populate our editor with the SQL query:

```
SELECT
    eventSource, eventName, COUNT(*) AS apiCount
FROM
    f259f21e-1018-4863-9d63-52b08844b84d
WHERE
    eventTime > '2023-11-20 00:00:00'
GROUP
    BY eventSource, eventName
ORDER
    BY apiCount DESC
```

Click on `Run` and give it a few seconds. By default, you'll see the `Command output` tab, but let's switch over to the `Query results` tab.

In the `Results`, you'll be able to see the `eventSource`, `eventName`, and `apiCount`, and it will be ordered from most API calls to least, which gives you a useful view to understand what API calls are being made the most in your AWS account.

Sample query #2

Let's go back to `Sample queries` and this time let's search for `Investigate manually created resources`. Click on that and you'll see this query:

```
SELECT
    userIdentity.arn AS user, userIdentity, eventTime,
    eventSource, eventName, awsRegion, requestParameters,
    resources, requestID, eventID
FROM
    f259f21e-1018-4863-9d63-52b08844b84d
WHERE
    (eventName LIKE '%Create%')
    AND resources IS NOT NULL
    AND userIdentity.sessioncontext.sessionissuer.username NOT LIKE 'AWSServiceRole%'
    AND userIdentity.sessioncontext.sessionissuer.username IS NOT NULL
    AND sourceIpAddress != 'cloudformation.amazonaws.com'
```

```
    AND eventTime > '2023-11-20 00:00:00'
ORDER
    BY eventTime DESC
```

Before we run this query, let's take a closer look at how they work.

We have a `SELECT` statement where we're selecting:

- `userIdentity.arn AS user`
- `userIdentity`
- `eventTime`
- `eventSource`
- `eventName`
- `awsRegion`
- `requestParameters`
- `resources`
- `requestID`
- `eventID`

These are the `Event properties` that you can also see and search for in the left bar of this editor. This is one of the very valuable features we get from Lake, because we can include multiple properties in both our `SELECT` statement and in the `WHERE` clause to refine our search.

Next we have our `Event data store ID` in the `FROM` clause.

Then we have:

- Our `WHERE` statement which is filtering for `eventNames` that include the word `Create` in them
- and `resources IS NOT NULL` meaning that the resources property needs to have a value
- and `userIdentity.sessioncontext.sessionissuer.username NOT LIKE 'AWSServiceRole%'` which is checking to see that the resource was not created by a service-linked role
- and `userIdentity.sessioncontext.sessionissuer.username IS NOT NULL`
- and `sourceIpAddress != 'cloudformation.amazonaws.com'` which is the part that checks for whether the resource was created manually or not — keep in mind that this could produce false positives if you're not using CloudFormation but you're using some other sort of automation to deploy resources; of course you can modify this query, this is just a starting point
- and an `eventTime` filter

Finally, we order the results by `eventTime` in descending order.

Even if you're not familiar with SQL and querying databases, you can see that this is fairly straight forward and mostly self explanatory…but if you have any questions, don't hesitate to ask!

OK, let's `Run` this query!

While we wait for the results, I'll point out that you can also check the box that says `Save results to S3` if you'd like to keep a long-term record of this result.

Our query successfully ran, so let's click on the `Query results` tab.

We'll see a couple of results here, and we can view exactly who made what call, against which service, etc…

In this case, I created KMS grants in the `us-east-1` region.

This query is practical for a few use cases, including to see if anyone is creating resources outside of the organization's automation tooling, which often leads to issues down the road, including creating shadow IT resources.

This query could also help from a security perspective because if your account becomes compromised, it's entirely possible that the attacker will create resources "manually" and not through CloudFormation or whatever other automation you're querying for. You'd be able to see those resources and what was created with this query.

Conclusion and deleting the event data store

As you can see, we're scratching the surface of what can be done with CloudTrail Lake, but hopefully this gave you a great starting point.

Feel free to keep playing around with this feature, and then I recommend deleting your event data store to prevent additional charges, unless you plan on keeping this for future use!

Go back to `Event data stores` from the `Lake` menu, then click on the name of your event data store, click on the `Actions` dropdown and `Change termination protection` to select `Disabled` and then `Save`.

You can now click on `Actions` again and this time click on `Delete`.

CloudTrail will keep this visible for 7 days, after which it gets permanently deleted. During that time, you can restore it if you made a mistake or changed your mind.

But that's it! Congrats on creating your first event data store and running queries against it!

Go ahead and complete this lesson, and I'll see you in the next!

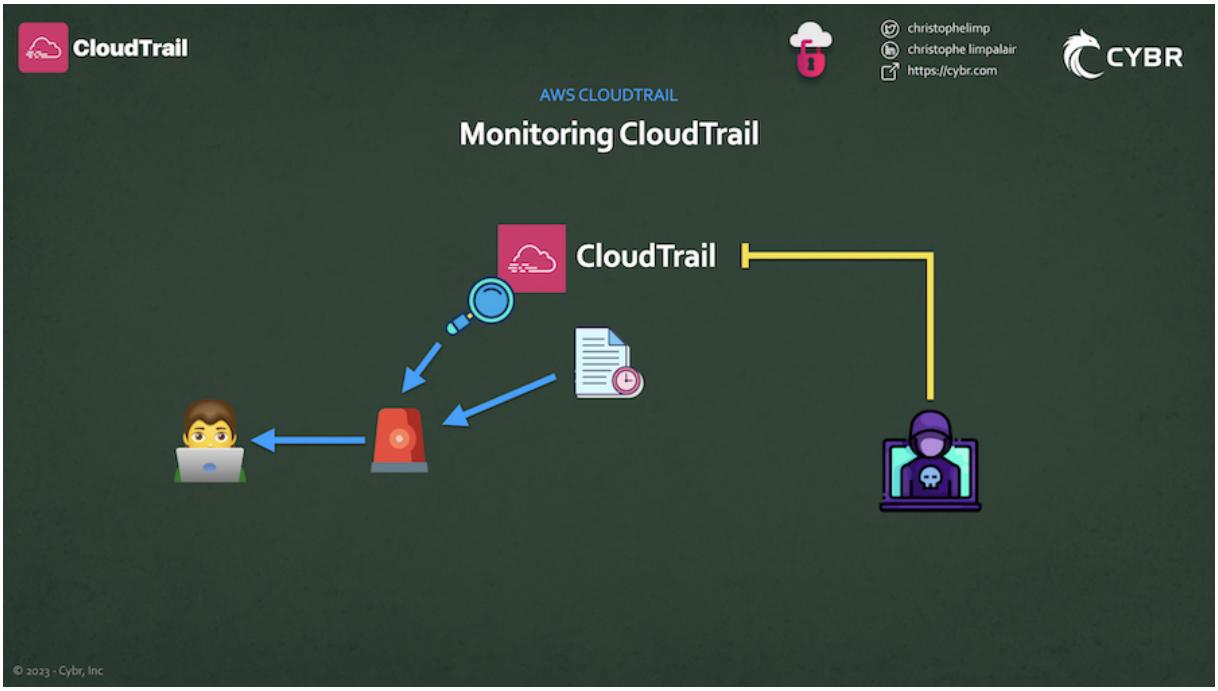
Security and Best Practices

Monitor CloudTrail itself

The first topic I'd like to talk about when it comes to protecting our CloudTrail deployment is about monitoring CloudTrail itself.

It's important to monitor the CloudTrail service and keep a log of changes made to it. Attackers always want to hide their tracks, so they may attempt to stop the CloudTrail service or to make other modifications. If that ever happens, you need to not only get notified, but you also need to keep track of those changes.

Let's start by talking about what you should be monitoring at a minimum, and then we'll talk about some ways of implementing that.



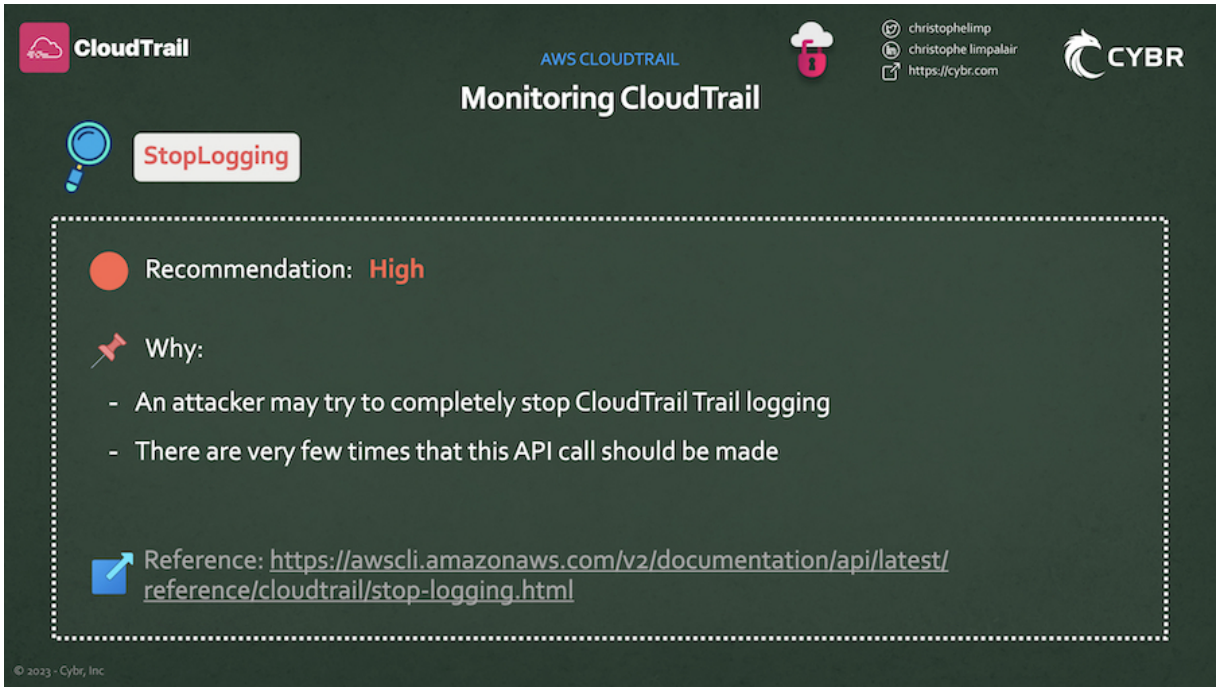
StopLogging

Reference: https://awscli.amazonaws.com/v2/documentation/api/latest/reference/cloudtrail/stop-logging.html

The first API call that we should be monitoring is the `StopLogging` call. if an attacker gets ahold of credentials and starts making API calls against your AWS environments, after doing some simple enumeration, they may try to completely stop CloudTrail from logging data.

One way they can do that is with the `StopLogging` call.

There are very few times that you should be calling this API, so monitoring this should not result in alert fatigue and is definitely one I would recommend keeping an eye on.



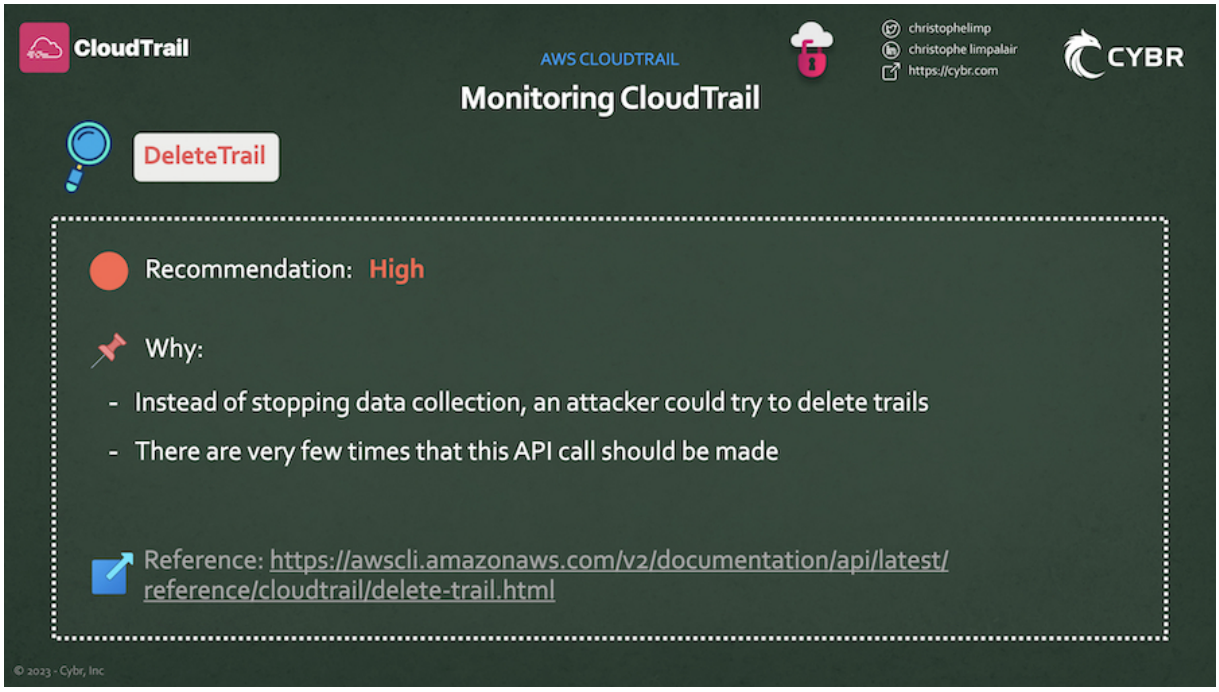
DeleteTrail

Reference: https://awscli.amazonaws.com/v2/documentation/api/latest/reference/cloudtrail/delete-trail.html

The next call I'd recommend keeping an eye on is the `DeleteTrail` call.

Instead of just trying to stop a trail from logging, an attacker may try to completely delete it.

Again, this is an API call you most likely won't be using very much if at all, and so it's the next one I would watch for.



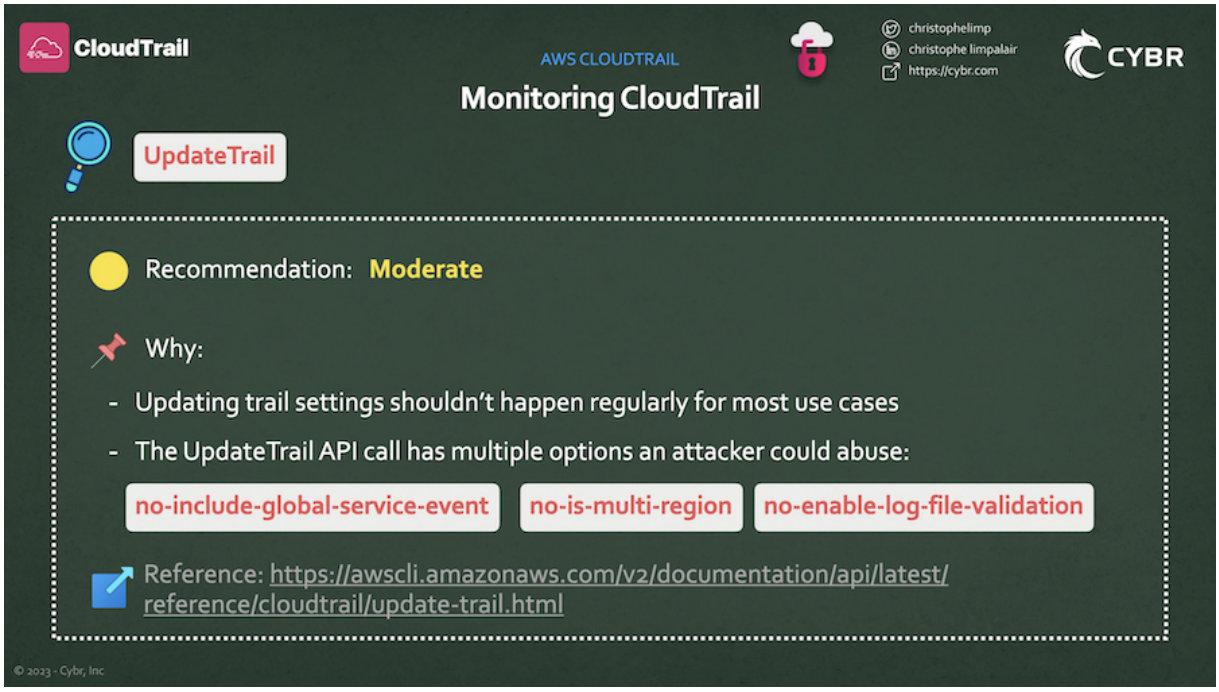
UpdateTrail

Reference: https://awscli.amazonaws.com/v2/documentation/api/latest/reference/cloudtrail/update-trail.html

Updating trail settings is something that you shouldn't be doing regularly. Typically, this happens almost never once you've set it up, apart from the initial setup and an occasional change in settings.

So it's not a bad idea to monitor the `UpdateTrail` API call to be aware of any changes made to your trails, especially because there are a few options you can pass in to the `UpdateTrail` call that would give cause for concern from a security perspective:

- `no-include-global-service-event` – which controls whether CloudTrail is collecting events from global services like IAM
- `no-is-multi-region` – which would let an attacker disable multi-Region logging and only enable a single region
- `no-enable-log-file-validation` – which would let an attacker disable log file validation

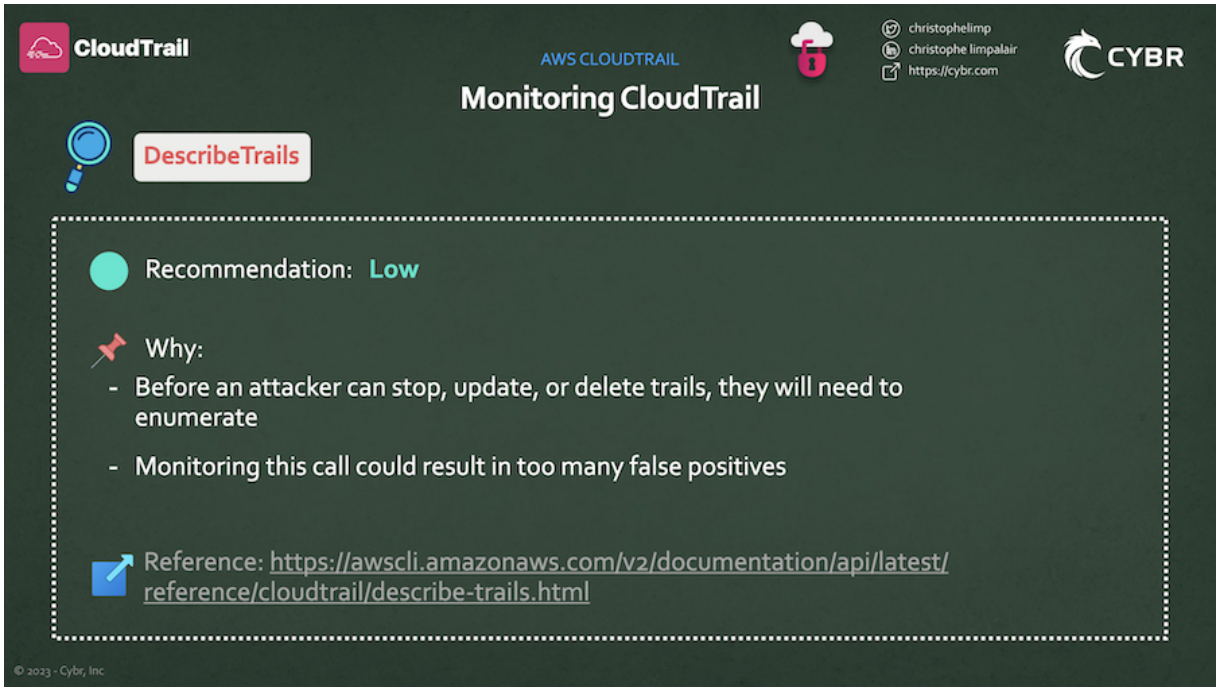


DescribeTrails

Reference: https://awscli.amazonaws.com/v2/documentation/api/latest/reference/cloudtrail/describe-trails.html

Before deleting or stopping trail logging, an attacker will need to know what trails, if any, are enabled for an AWS account. They can do that with the `DescribeTrails` API call, so that could be one you keep an eye on.

This one could lead to generating too many false alerts, so it's up to you, but it could be a good one to monitor.



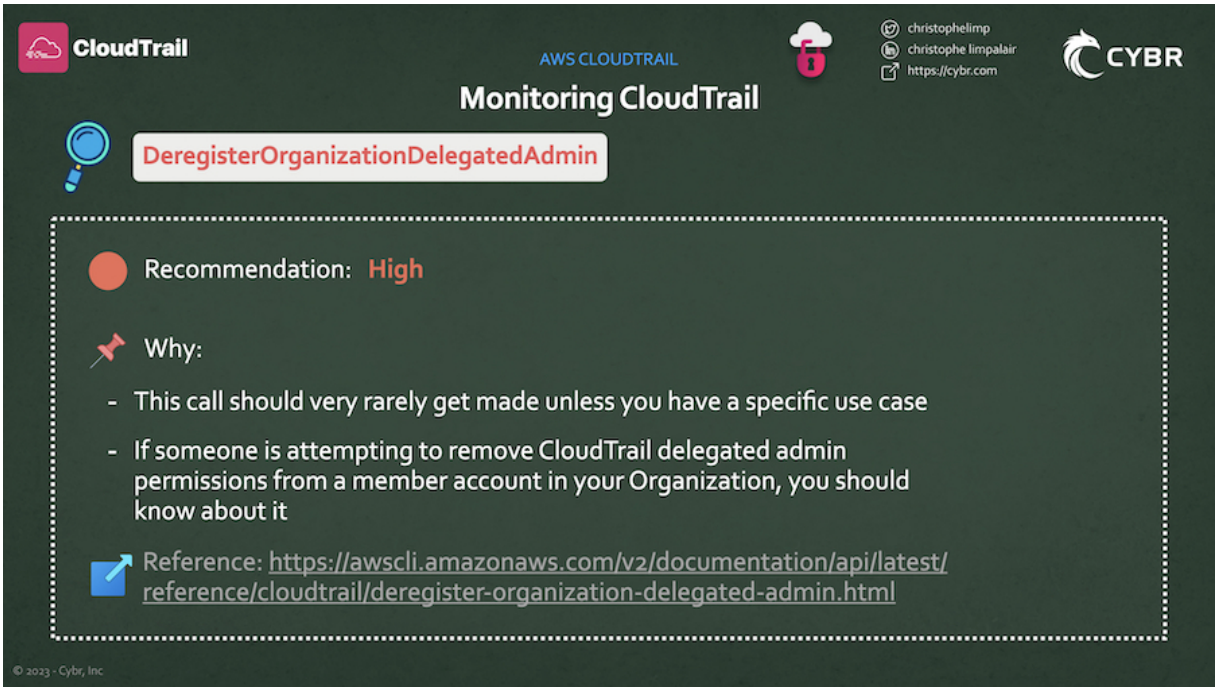
DeregisterOrganizationDelegatedAdmin

Reference: https://awscli.amazonaws.com/v2/documentation/api/latest/reference/cloudtrail/deregister-organization-delegated-admin.html

As a good practice, you should also keep an eye on the call `DeregisterOrganizationDelegatedAdmin` because this should rarely be happening, especially in your most critical AWS accounts.

If someone is attempting to remove CloudTrail delegated administrator permissions from a member account in an organization, you should definitely be aware of the attempt.

If you're not familiar with delegated administration, this is a useful feature for multi-account setups.



How to monitor for these calls?

Great, so we know there are API calls we should monitor, but how exactly do we do that? You've got a few options, but I'll mention 2 to get you started:

- Using the AWS SSK
- CloudWatch Metrics and Alarms

Using the AWS SSK

The AWS SSK is something you might be familiar with if you took my Introduction to AWS Security course, but if not, it's a free and open source project designed to help deploy the bare minimum security monitoring and alerting you need for your AWS accounts.

As part of that, it deploys monitoring and alerting for:

- `StopLogging`
- `DeleteTrail`
- `UpdateTrail`

The way that it does this is by using EventBridge Rules, as you can see here.

So by deploying this tool, it would take care of monitoring and alerting for those 3 API calls, plus a whole lot more that's not necessarily related to CloudTrail. You could then make modifications to the tool in order to monitor and alert for those other API calls I mentioned. Since this uses EventBridge rules, you have a lot of flexibility here.

CloudWatch Metrics and Alarms

Otherwise, if you want something simpler and you don't want to deploy the SSK solution, you can build your own alarms from scratch. You can either do that with EventBridge just like the SSK did, or you can do that using CloudWatch Metrics and alarms.

Let's take a look at an example.

There are a couple of ways you can do this, but for this example, let's go over to `Logs` and then `Log Groups`. Select the CloudTrail log group that you created for this course, and then click on the `Metric filters` tab. From there, click on `Create metric filter`.

You have a lot of options for the `Filter pattern`. Let's use this pattern:

```
{ ($.eventName = UpdateTrail) || ($.eventName = DeleteTrail) || ($.eventName = StopLogging) }

Code language: JavaScript (javascript)
```

You can then test this pattern against fake data or real data from that specific log group, which is really practical.

Click on `Next`.

Give it a name, like `CloudTrailChanges`, a `Metric namespace` of `CloudTrailMetrics`, a `Metric name` of `CloudTrailChangesEventCount` and a `Metric value` of `1`.

Click `Next`, review, and `Create metric filter`.

We then need to create an alarm for this. Select the metric you just created by clicking on its checkbox, and then click on the `Create alarm` button.

You'll want to keep the `Statistic` set to `Sum`. Further down you will want to keep the `Threshold type` to `Static`. You can also keep it set to `Greater` and you can then set the threshold value to `0`. Or you can select `Greater/Equal` and set the value to `1`. Up to you.

On the next page, we want to make sure it's set to `In alarm`.

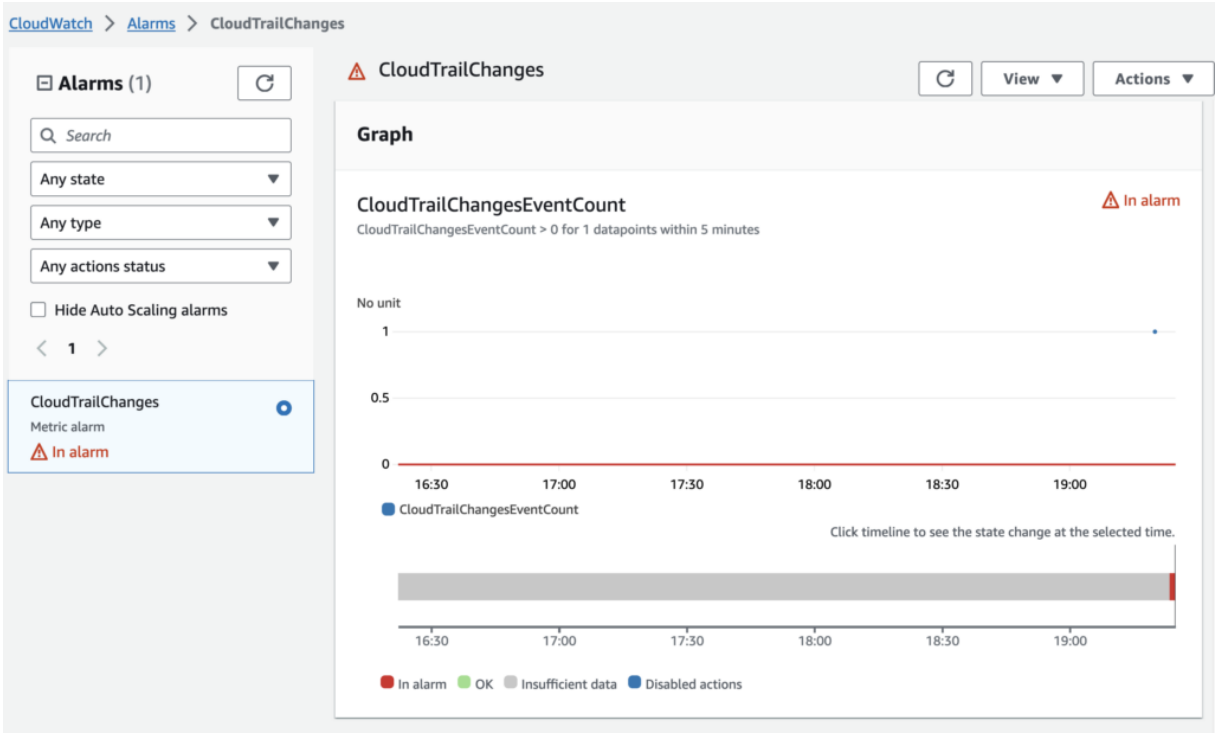
You can then select how to send your SNS notification by `Selecting an existing SNS topic` if you have one (which I do so that's what I'll select), or `Creating new topic`.

On the next page, I'll name my alarm `CloudTrailChanges`. Click on `Next`, review settings, and `Create alarm`.

Now to test this alarm, let's go over to the CloudTrail Dashboard, and let's update it by stopping logging on the trail that you created for this course. Select the trail, and click on the `Stop logging` button.

You'll need to give CloudWatch about 5 minutes, but you should then receive a notification and you should be able to click on your new alarm in CloudWatch and refresh the graph for data points to populate. If you don't yet, just give it a few minutes.

I just received an email alarm notification, and if we refresh this page we will see the data point in the graph and that it is in the alarm state.



Conclusion

You now have a few options in your toolkit to monitor for important CloudTrail API calls.

Let’s complete this lesson and let’s talk about a few more security best practices for protecting our security logs.

IAM

If you’re not already familiar with it, AWS Identity and Access Management (or IAM for short), is the service that lets you control who can be authenticated and authorized to use your AWS resources, including CloudTrail resources.

If you don’t get your IAM configurations right, then you will leave your CloudTrail deployments vulnerable. So we need to start there when it comes to securing CloudTrail.

Because IAM is such a large and complex service, we’ll reserve explaining that in more detail for a different course, but for this lesson, let’s focus on what you need to know and do for CloudTrail specifically.

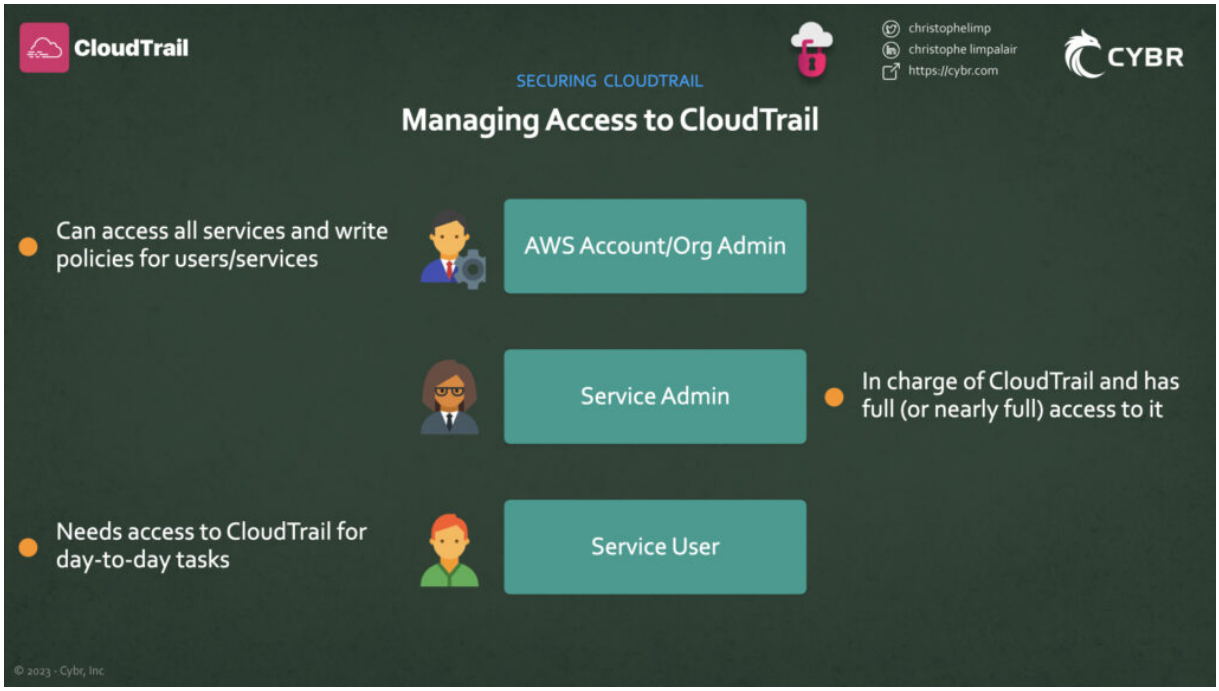
Managing access to the CloudTrail service

To get started, let's talk about managing access to the CloudTrail service itself.

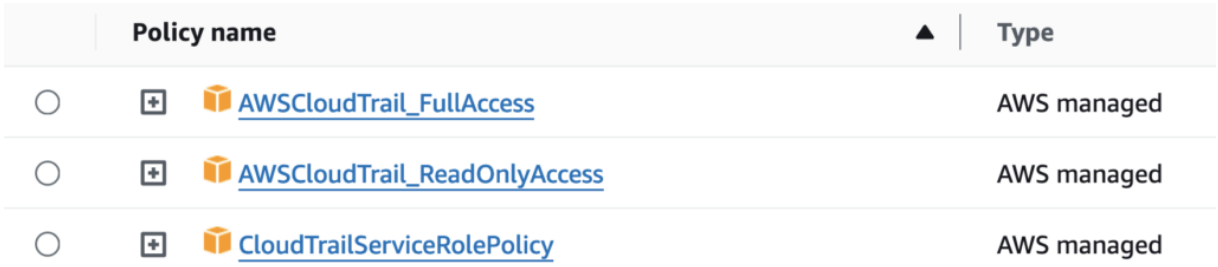
At the highest level, you will have your AWS admins who can access most or all services and write policies for other users or services.

You will then have service administrators, who are in charge of CloudTrail resources at your company, and so they will likely have full access (or close it) to the CloudTrail resources.

Then you have service users, who may need to use the CloudTrail service to do their day-to-day jobs, and so the administrator will need to provide them with the least privilege permissions.



When creating policies to grant access to those different types of users, a helpful starting point is to use AWS-managed policies, and then modify them to your own custom use case by providing less access.



As you can see, we currently have 3 managed policies including:

- Full Access
- Read Only Access
- Service Role

For service admins, you could start with full access and scale back as needed.

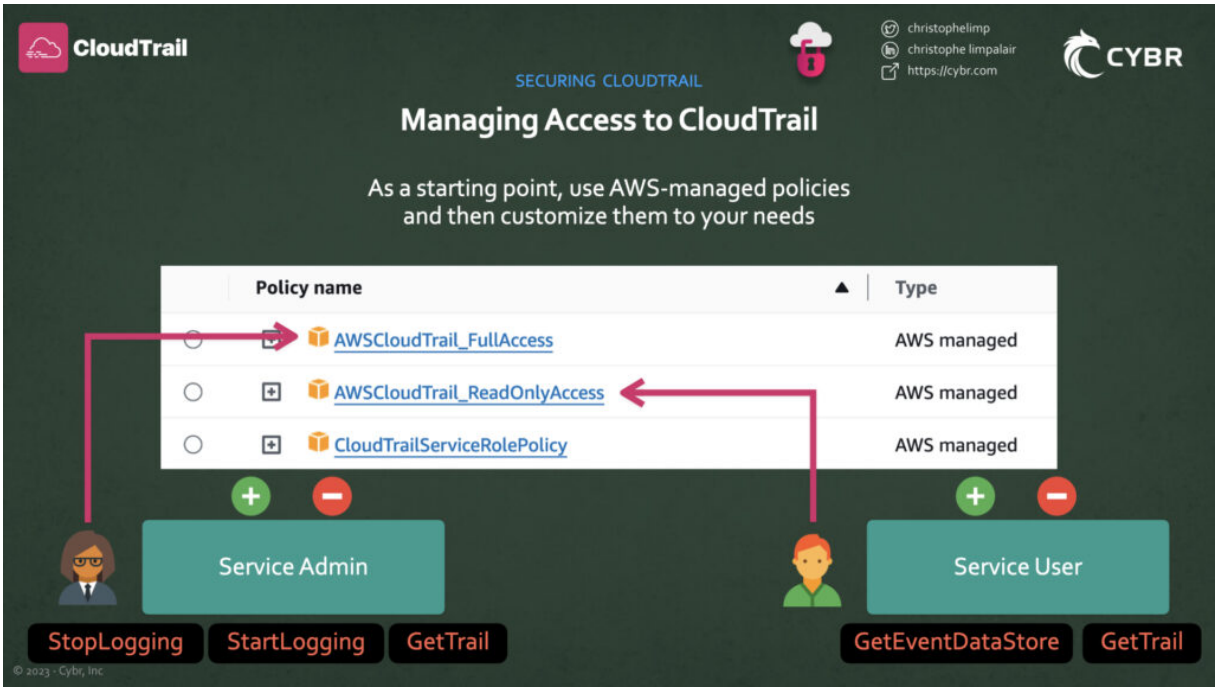
For service users, you could start with Read Only Access and add or remove as needed.

It's typically better to restrict more than you think will be needed, and then to have the users request more access as they run into permissions issues, until you get to the right policy.

For example, your service administrators may need permissions to `StopLogging`, `StartLogging`, `GetTrail`, etc…

But a service user most likely shouldn't have access to `StopLogging`. That's just not usually something a service user needs to do to complete their tasks, so there's no reason to give them that permission.

That service user, however, may need to be able to use CloudTrail Lake in order to query for information, and so you could grant them access to specific event data stores.



Managing access to CloudTrail logs

Apart from securing the CloudTrail service itself, we also need to secure CloudTrail logs.

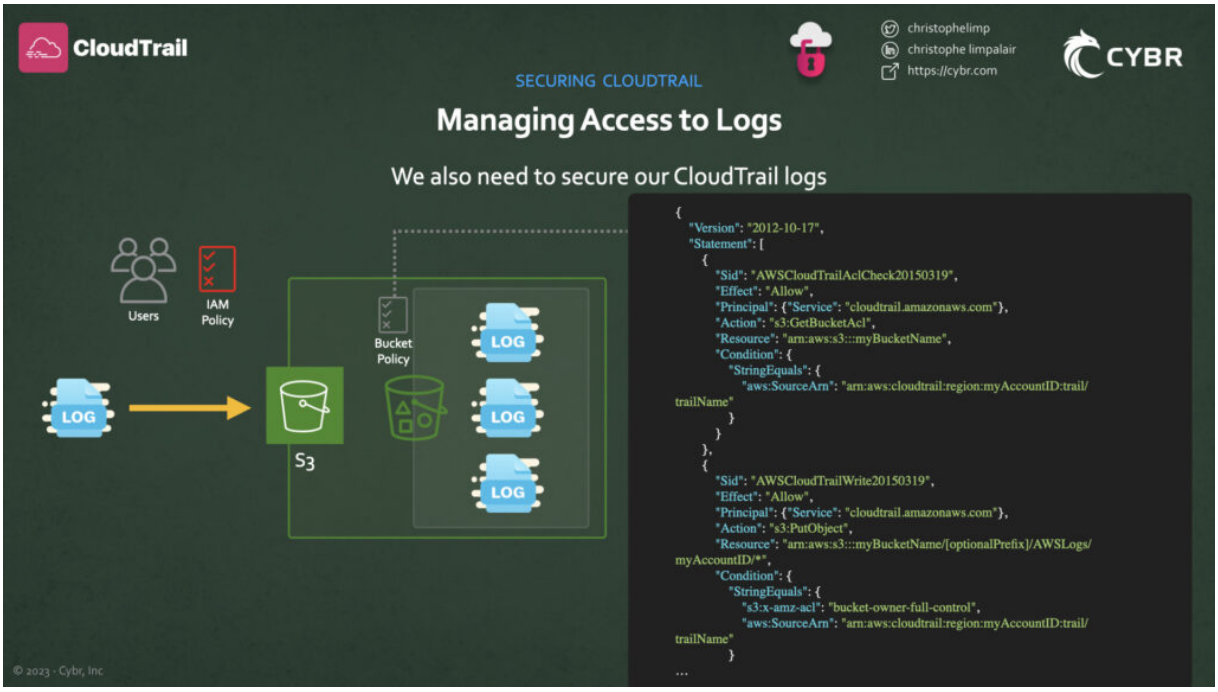
As we know, these get saved into Amazon S3, and so we can use both IAM policies, as well as S3 bucket policies, to control who can view, modify, or delete those logs.

As a best practice, you should be using a dedicated S3 bucket for your CloudTrail logs — you should not be mixing logs with other business data, for example. That makes securing the bucket a lot easier.

AWS recommends this S3 bucket policy:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AWSCloudTrailAclCheck20150319",
            "Effect": "Allow",
            "Principal": {"Service": "cloudtrail.amazonaws.com"},
            "Action": "s3:GetBucketAcl",
            "Resource": "arn:aws:s3:::myBucketName",
            "Condition": {
                "StringEquals": {
                    "aws:SourceArn": "arn:aws:cloudtrail:region:myAccountID:trail/trailName"
                }
            }
        },
        {
            "Sid": "AWSCloudTrailWrite20150319",
            "Effect": "Allow",
            "Principal": {"Service": "cloudtrail.amazonaws.com"},
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::myBucketName/[optionalPrefix]/AWSLogs/myAccountID/*",
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": "bucket-owner-full-control",
                    "aws:SourceArn": "arn:aws:cloudtrail:region:myAccountID:trail/trailName"
                }
            }
        }
    ]
}
```

Which gives permissions to the CloudTrail service to put objects into that S3 bucket.



This policy by itself, however, would not grant read permissions to other users, like service users. Maybe that's how you want to keep it, because you want to force service users to read the logs through Lake, the API, or CloudWatch Logs. But if a user needs to have read access directly in S3, you would have to add that permission to the bucket policy and potentially also to their user's IAM policy.

So that's something important to keep in mind.

Conclusion

As I mentioned, AWS is a complicated service to get right, and we only have so much time to cover it in this course. For more in-depth information, please refer to this documentation.

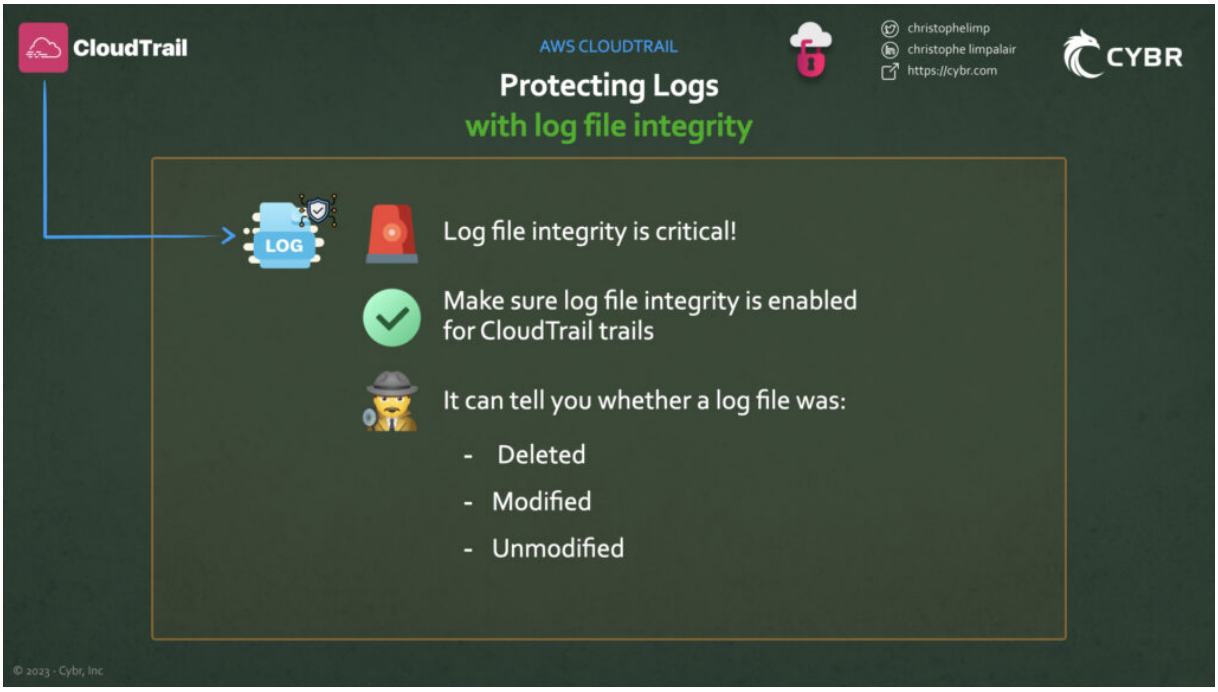
Once you're ready, go ahead and complete this lesson, and I'll see you in the next!

Log file integrity

The integrity of logs is extremely important. If you can't trust that the logs you are looking at are legitimate logs, then they become nearly useless.

It's highly recommended that you enable CloudTrail log file integrity for any existing trails you have, but this is a feature that's enabled by default for new trails.

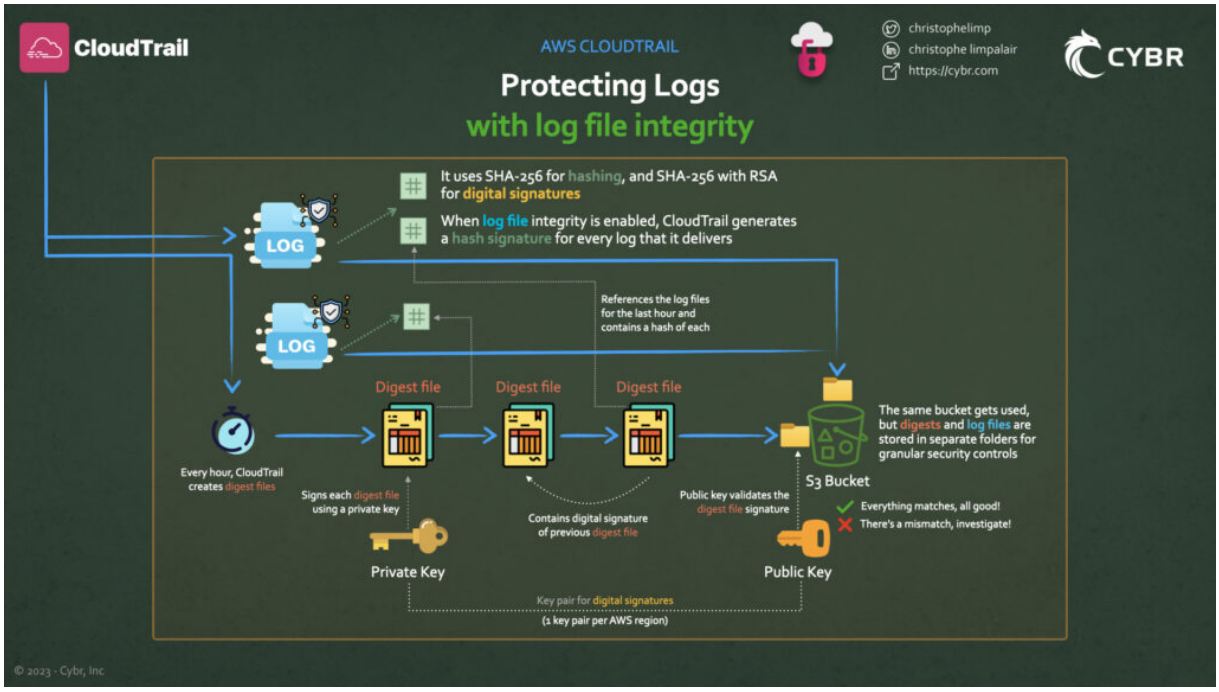
By having it enabled, you can tell whether a log file was deleted, modified, or you can confidently say that it was unmodified since it was delivered to S3.



It works by using SHA-256 for hashing, and SHA-256 with RSA for digital signatures.

With log file integrity, CloudTrail creates a hash for every log file that it delivers, and then every hour, it also creates a digest file that contains a hash of each of the log files, which you can use to compare and make sure that the integrity of the log files is intact.

CloudTrail also uses a private key that corresponds with a public key to sign each of the digest files in order to create a digital signature. This means you can use the public key to validate the digest files and make sure that everything checks out.



It is important to keep in mind that when you enable this log file validation feature, all you are doing is telling CloudTrail to generate the digest files we just talked about…it is *not* validating the integrity of the log files. You have to do that either through the AWS CLI using the `validate-logs` command, by using a third-party solution, or by creating your own custom solution. So this is not a solution you enable and then just assume it means your files haven't been tampered with!

Conclusion

As we wrap up, I recommend leaving log file validation enabled when you create new trails, and then I recommend using the `ValidateLogs` API call whenever you need to investigate a potential security incident just to make sure that what you're looking at is legitimate.

Go ahead and complete this lesson, and I'll see you in the next!

Encryption

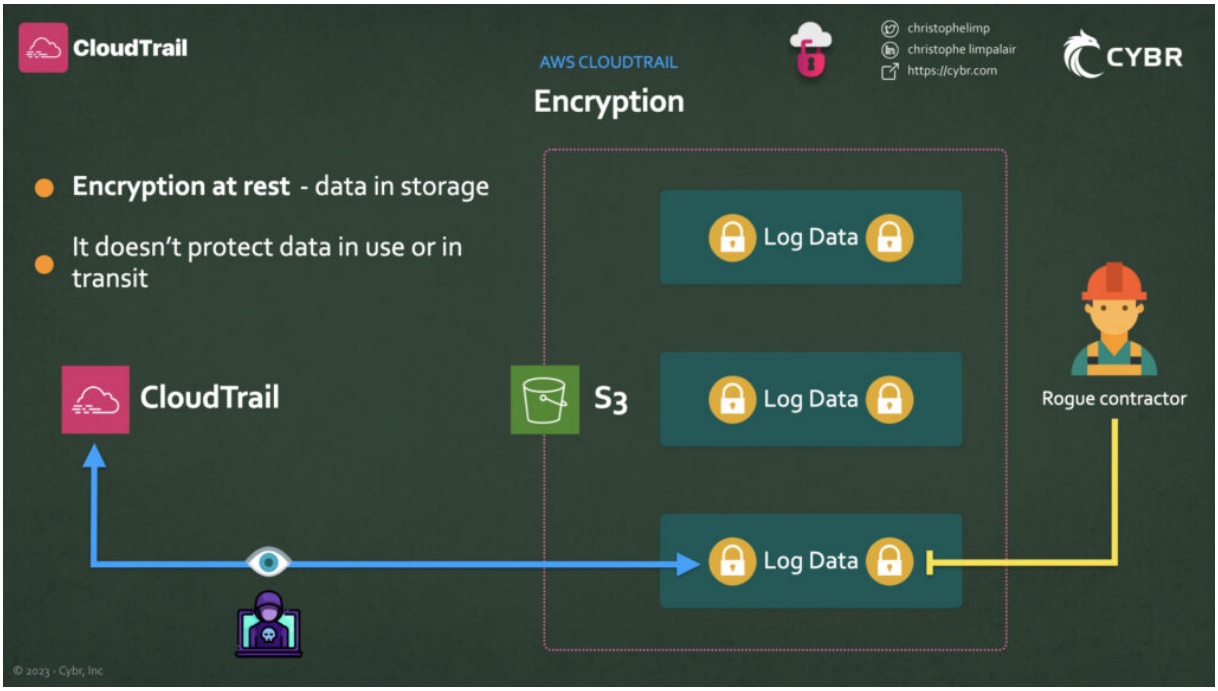
CloudTrail encryption is relatively straightforward so we'll keep this lesson simple, but there are a couple of key concepts to understand.

Encryption at rest

When people are talking about CloudTrail encryption, they are usually talking about encryption for data at-rest, which means the data that's stored on AWS' servers in their data centers.

This type of encryption is important because it helps protect sensitive data from AWS employees being able to access it and see what it is in plaintext, and/or if anyone ever steals the storage drives from those data centers or as they are being discarded after their end of life.

However, by default, it doesn't protect the data beyond that.



Whenever you are viewing the data in the AWS console or via the API, that data will be decrypted and in plaintext. Or if you download it to a local device, that encryption at rest won't be what makes sure the data is downloaded in an encrypted format.

By default, log files delivered by CloudTrail to S3 are encrypted by server-side encryption using Amazon S3-managed encryption keys. This is referred to as SSE-S3.

AWS instead recommends that you enable SSE-KMS, which is short for server-side encryption using AWS KMS keys. That way, you are in direct control of the encryption keys being used.

Enabling SSE-KMS

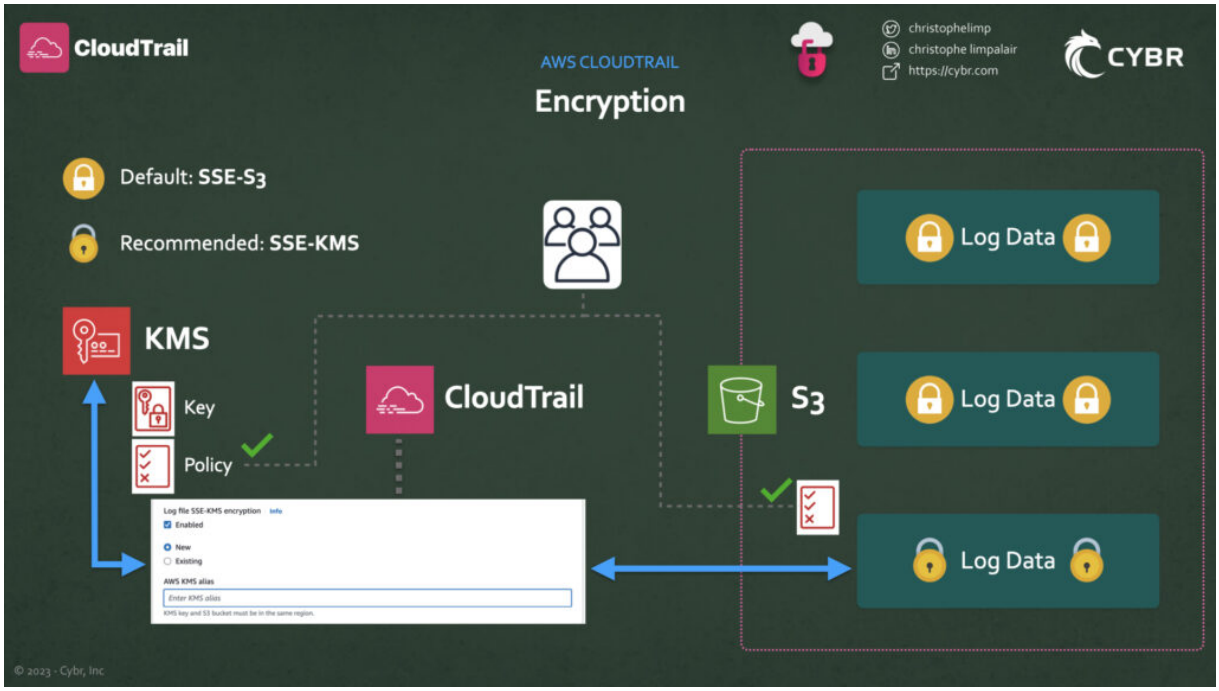
To use this, you would first create a KMS key from the AWS KMS service, and you can configure CloudTrail to use that key when you are creating a trail, or after the fact when you edit the trail.

You just need to make sure that the key and S3 bucket reside in the same region.

KMS keys have policies attached to them that determine who can use the key for encrypting and decrypting, and as long as the user is authorized, the encryption and decryption process is completely transparent to them.

This is one of the main benefits of using SSE-KMS, because now not only does a user have to have S3 read permissions for the bucket that contains the log files in order to access CloudTrail log files (like you would expect), but that user *also* needs to have a policy or role applied that allows decrypt permissions by the KMS key policy.

So we now have an extra layer of security.



Conclusion

So that's it, I just wanted to quickly cover why you would want to consider using SSE-KMS, and a brief explanation of how you would enable that.

Complete this lesson, and I'll see you in the next!

[Checklist] Security Hub Controls for CloudTrail

Throughout this course, we've been talking about best practices for using CloudTrail and securing our trails and data.

Understanding these best practices and implementing them in production can be two very different things — especially when you are busy with your day-to-day responsibilities, and as your AWS environments grow in size and complexity.

To help make sure that we follow these best practices let's take a look at a checklist.

I created this checklist by taking what AWS Security Hub is doing behind the scenes and putting it in a cheat sheet.

Security Hub, if you're not familiar, is a service we can enable to assess our AWS environments against security industry standards and best practices.

So this service will collect data across your accounts and it will identify and prioritize security issues that it finds.

7 potential issues with CloudTrail deployments

In this case, by default, Security Hub looks for 7 potential issues with CloudTrail deployments:

- [CloudTrail.1] CloudTrail should be enabled and configured with at least one multi-Region trail that includes read and write management events
- [CloudTrail.2] CloudTrail should have encryption at-rest enabled
- [CloudTrail.3] CloudTrail should be enabled
- [CloudTrail.4] CloudTrail log file validation should be enabled
- [CloudTrail.5] CloudTrail trails should be integrated with Amazon CloudWatch Logs
- [CloudTrail.6] Ensure the S3 bucket used to store CloudTrail logs is not publicly accessible
- [CloudTrail.7] Ensure S3 bucket access logging is enabled on the CloudTrail S3 bucket

Using this cheat sheet that I created and that you can download, you can keep this as a reference check to validate your CloudTrail deployments, and you can use this information to set up monitoring and alerting.

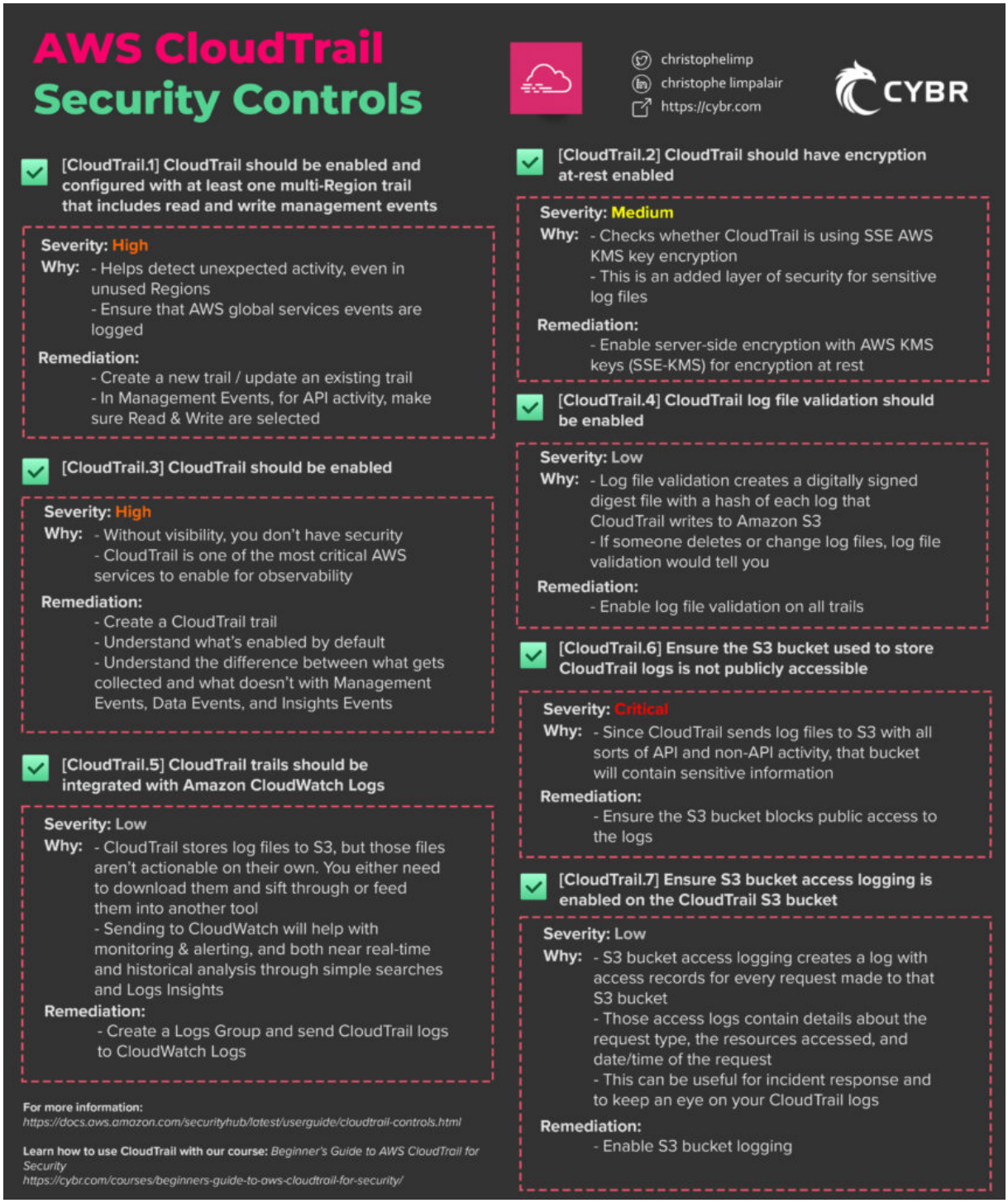


Image link: https://cybr.com/wp-content/uploads/2023/11/cloudtrail-security-controls-v2-1716x2048.jpg

Let's briefly walk through each of these one by one.

[CloudTrail.1] CloudTrail should be enabled and configured with at least one multi-Region trail that includes read and write

management events

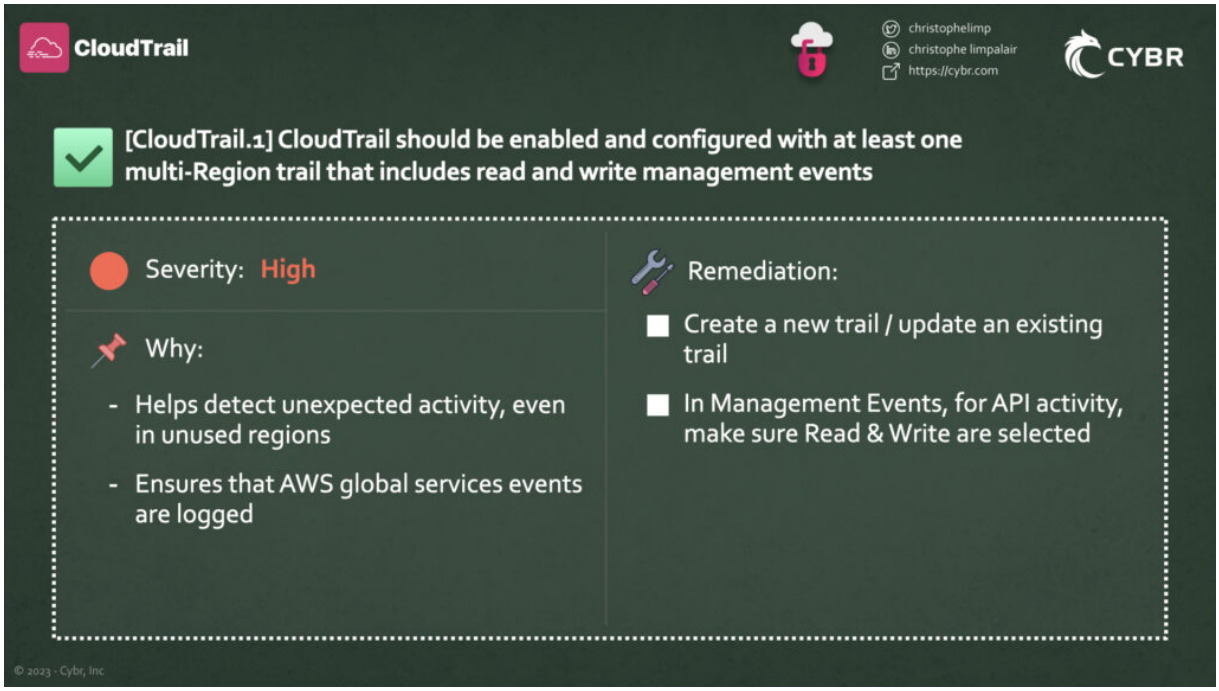
🟠 Severity: High

📌 Why:

- Helps detect unexpected activity, even in unused Regions – Ensures that AWS global services events are logged

🔐 Remediation:

- Create a new trail / update an existing trail
- In Management Events, for API activity, make sure Read & Write are selected



The severity of this finding is high, because if you don't have CloudTrail enabled for multi-region and for read/write management events, then you could have a threat actor carrying out an attack in a region that you never use and therefore never monitor, and you would have no idea that it's going on.

Enabling this also helps ensure that AWS global services events are logged, since some of AWS' services operate globally and not per-region.

To remediate this finding, you would create a new trail or update an existing trail to be multi-region, and you would check in Management Events that Read & Write are selected for API activity.

[CloudTrail.2] CloudTrail should have encryption at-rest enabled

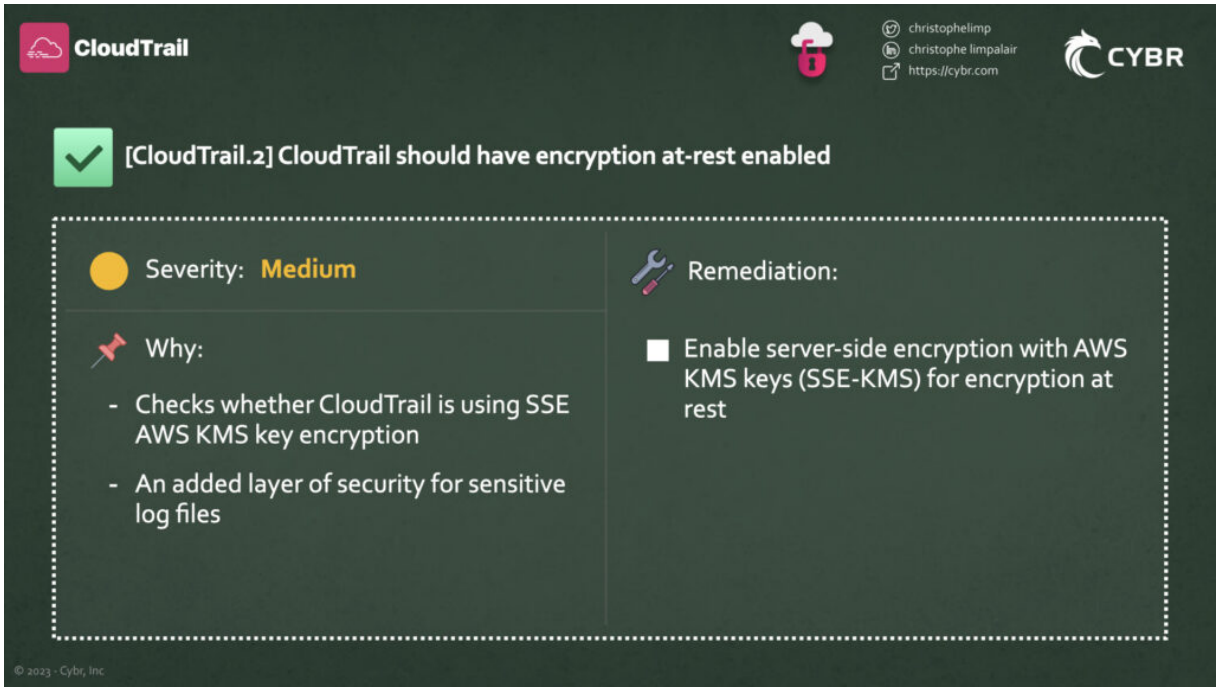
🟤 Severity: Medium

📌 Why:

- Checks whether CloudTrail is using SSE AWS KMS key encryption
- An added layer of security for sensitive log files

🔐 Remediation:

- Enable server-side encryption with AWS KMS keys (SSE-KMS) for encryption at rest



The severity of this finding is medium, because by default, AWS already provides some encryption at rest for CloudTrail log data.

However, they do recommend that you enable SSE-KMS, which stands for server-side encryption with AWS KMS keys, for an added layer of security.

Keep in mind that this encryption really only helps at rest, meaning it will protect against physical access to the data, like if an AWS employee went rogue, but it will not help prevent attackers from accessing your log data through the cloud. That's why we have other controls for that.

[CloudTrail.3] CloudTrail should be enabled

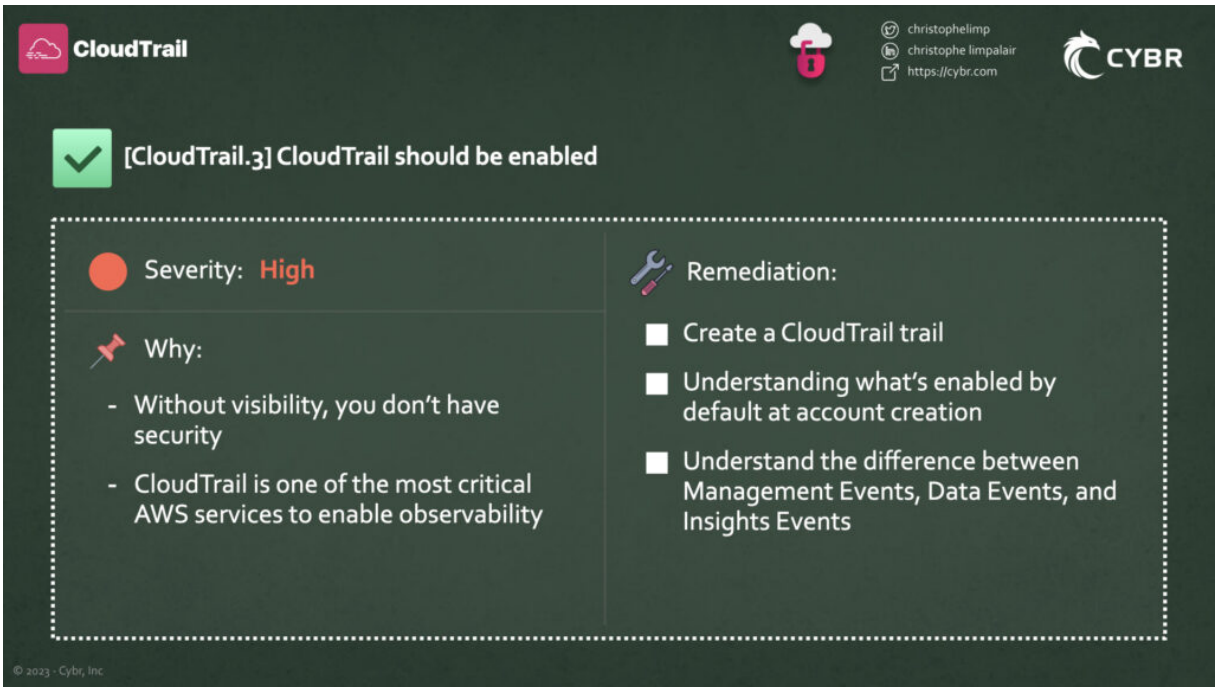
🟠 Severity: High

📌 Why:

- Without visibility, you don't have security
- CloudTrail is one of the most critical AWS services to enable observability

🔐 Remediation:

- Create a CloudTrail trail
- Understand what's enabled by default at account creation
- Understand the difference between Management Events, Data Events, and Insights Events



This finding has a severity of high because if you don't have CloudTrail enabled, you have little to no visibility into your AWS environments and resources. Personally I think this should be a Critical finding.

To remediate, you would go in and create a trail like I should you.

I also add in my cheat sheet that it's important to understand what's enabled by default at account creation and what the difference is between Management Events, Data Events, and Insights Events which you now fully understand because you've gone through this course!

[CloudTrail.4] CloudTrail log file validation should be enabled

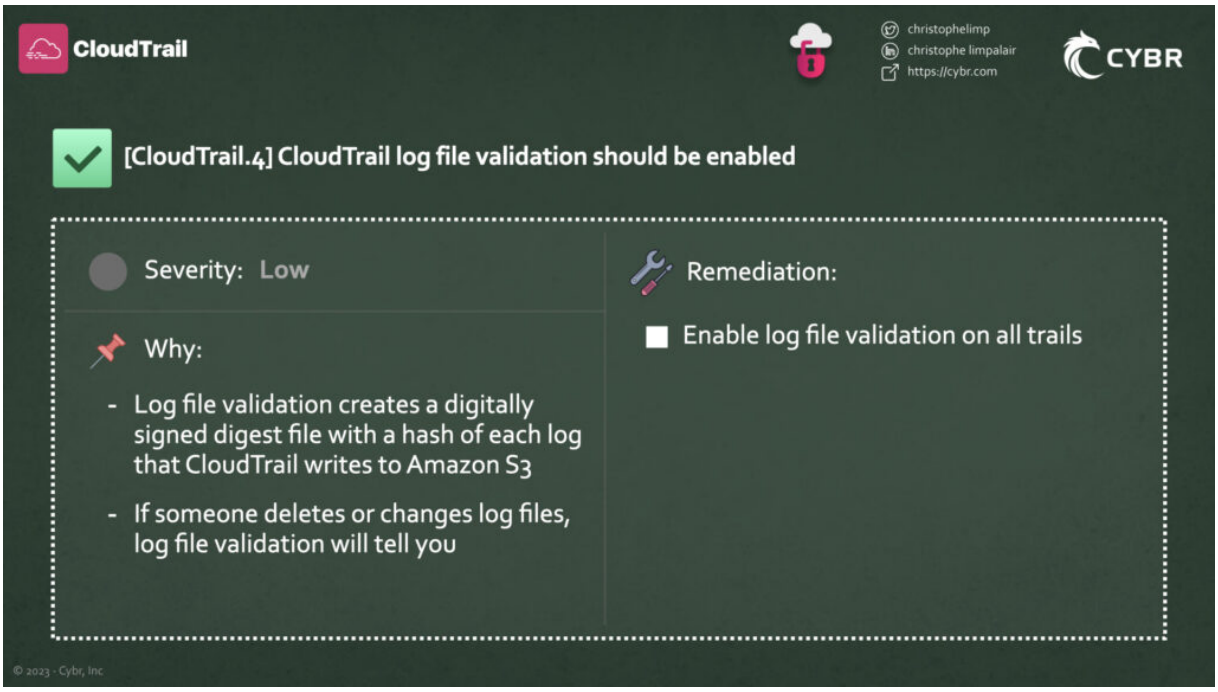
⚪ Severity: Low

📌Why:

- Log file validation creates a digitally signed digest file with a hash of each log that CloudTrail writes to Amazon S3
- If someone deletes or changes log files, log file validation will tell you

🔐 Remediation:

- Enable log file validation on all trails



For CloudTrail.4, the severity level is low.

The reason is because you can implement security controls to protect your log files, making it very difficult for someone to delete or modify them, but it is still recommended that you enable log file validation.

This creates a digitally signed digest file with a hash of each log that CloudTrail writes to Amazon S3, which means you can compare those digest files and see if the log files have been tampered with.

You remediate this finding by enabling log file validation on all of your trails.

[CloudTrail.5] CloudTrail trails should be integrated with Amazon CloudWatch Logs

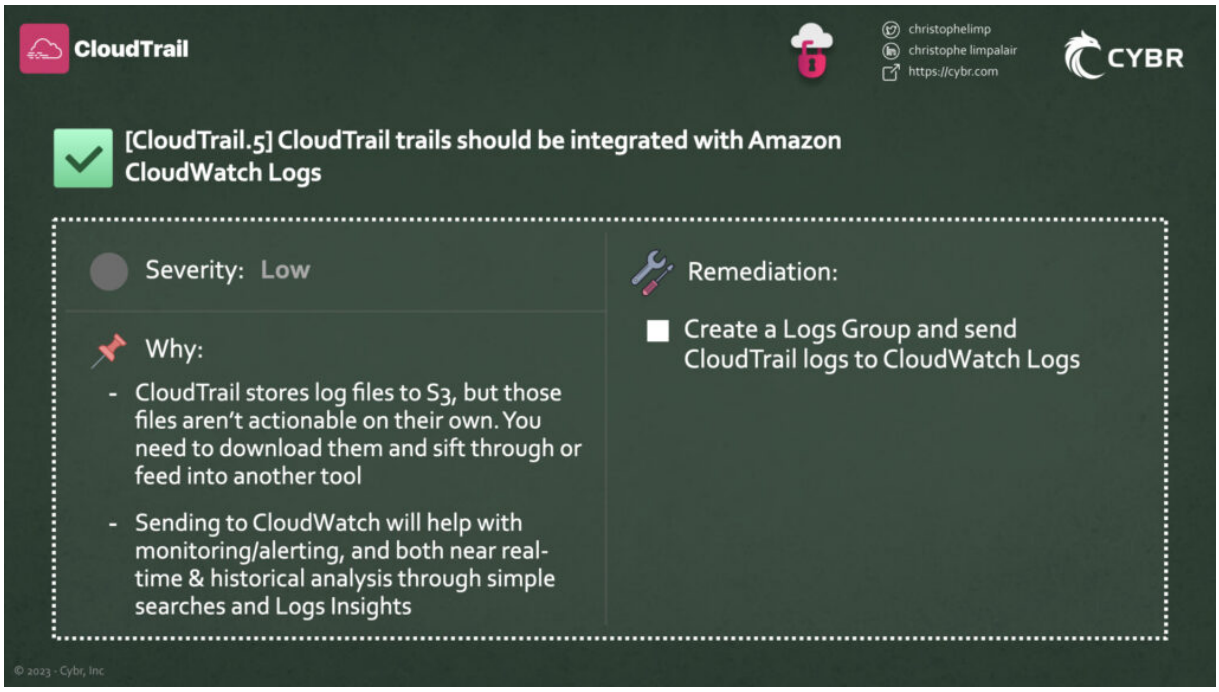
⚪ Severity: Low

📌Why:

- CloudTrail stores log files to S3, but those files aren't actionable on their own. You need to download them and sift through or feed into another tool
- Sending to CloudWatch will help with monitoring/alerting, and both near real-time & historical analysis through simple searches and Logs Insights

🔐 Remediation:

- Create a Logs Group and send CloudTrail logs to CloudWatch Logs



For this finding, we have a severity level of low. The reason it's low even though I recommend integrating with CloudWatch Logs, is because you can feed CloudTrail logs into other third-party solutions and still get the same result. You don't technically have to use CloudWatch Logs, but in my opinion, you should use at least one solution to feed your logs into, even if that doesn't end up being CloudWatch.

You remediate this by creating a CloudWatch Logs Group, and then editing your trail to send logs to that group.

[CloudTrail.6] Ensure the S3 bucket used to store CloudTrail logs is not publicly accessible

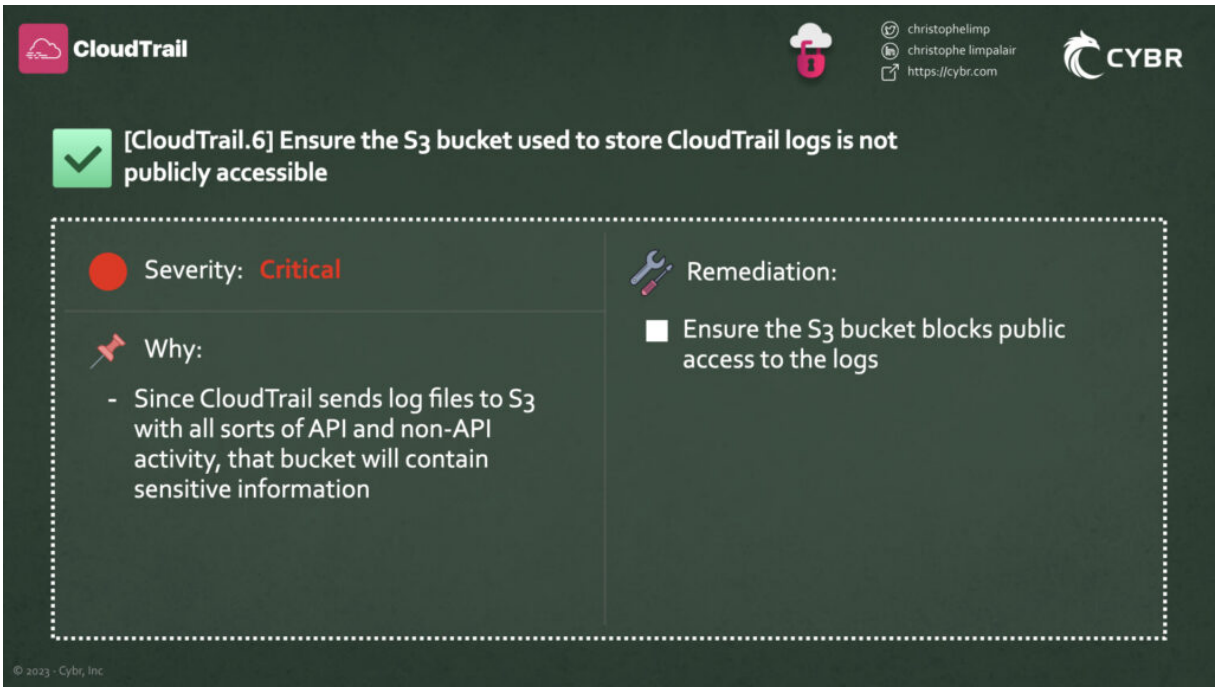
🔴 Severity: Critical

📌Why:

- Since CloudTrail sends log files to S3 with all sorts of API and non-API activity, that bucket will contain sensitive information

🔐 Remediation:

- Ensure the S3 bucket blocks public access to the logs



We then have a critical severity finding for ensuring that the S3 bucket storing your logs is not publicly accessible.

This one is fairly self explanatory — logs contain a bunch of API and non-API activity, which will contain sensitive data. That should never be publicly accessible.

To remediate, ensure that the S3 bucket blocks public access.

[CloudTrail.7] Ensure S3 bucket access logging is enabled on the CloudTrail S3 bucket

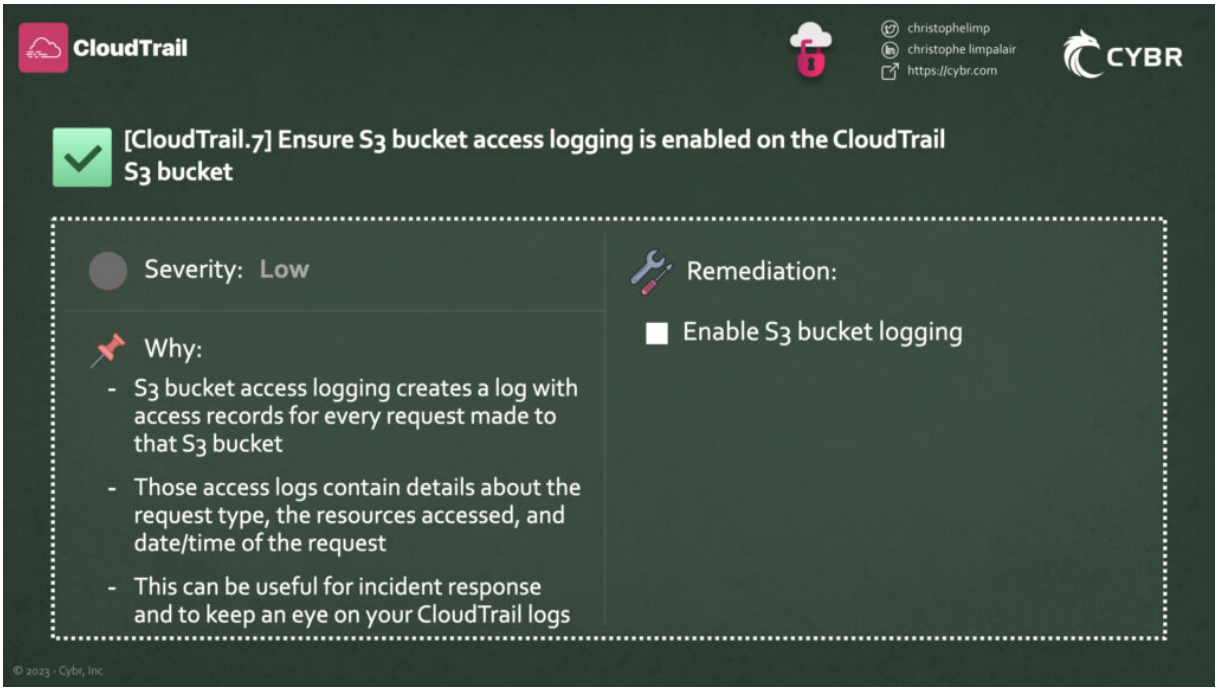
⚪ Severity: Low

📌Why:

- S3 bucket access logging creates a log with access records for every request made to that S3 bucket
- Those access logs contain details about the request type, the resources accessed, and date/time of the request
- This can be useful for incident response and to keep an eye on your CloudTrail logs

🔐 Remediation:

- Enable S3 bucket logging



Finally, we have a severity of low for CloudTrail.7.

This one is about ensuring that S3 bucket access logging is enabled.

Access logging creates a log with access records for every request made to that S3 bucket. Those access logs contain details about the request type, the resources accessed, and date/time of the request, so you can use this information for incident response and just to keep an eye on your logs.

Remediate by enabling S3 bucket logging.

Conclusion

So there you have it, you now have a checklist you can use to ensure that you are following CloudTrail security best practices.

Feel free to download this cheat sheet and share it with anyone who needs it!

Then, complete this lesson and I'll see you in the next!

Conclusion

[Cheat Sheet] Useful CloudTrail CLI command

Image link: https://cybr.com/wp-content/uploads/2023/11/cloudtrail-cli-cheatsheet-1716x2048.jpg

Note: This is not meant to be a comprehensive list of AWS CLI operations. For that, refer to this page. This is instead meant to be some of the most useful information for day-to-day.

General / Operational

```
# List trails in the account
aws cloudtrail list-trails

# Shows settings for one or more trails for the specified region (or current region if not specified)
aws cloudtrail describe-trails [--trail-name-list <value>] [--region <value>]

# Create a new trail
aws cloudtrail create-trail --name <value> --s3-new-bucket <value> [--include-global-service-events] [--
is-multi-region-trail] [--enable-log-file-validation] [--cloud-watch-logs-log-group-arn <value>] [--clou
d-watch-logs-role-arn <value>] [--kms-key-id <value>]

# List the names of all trails
aws cloudtrail describe-trails [--output json]

# Get the status of a trail
aws cloudtrail get-trail-status --name <value>

# Add tags to a trail, up to 10 tags
aws cloudtrail add-tags --resource-id <value> --tags-list "Key=log-events,Value=management"

# List the tags of a trail
aws cloudtrail list-tags --resource-id-list <value> # List of trail, event data store, or channel ARNs
```

Working with Trails

```
# Log file location
bucket_name/prefix_name/AWSLogs/Account ID/CloudTrail/region/YYYY/MM/DD/file_name.json.gz

# Example:
aws-cloudtrail-logs-212301213533-419e1363
AWSLogs/212301213533/CloudTrail/us-east-1/2023/11/10/10/272281913033_CloudTrail_us-east-1_20231110T2225Z
_LMThgKKp783oekGe.json.gz

# Search through log files manually
# Download jq (MacOS) or more instructions here: https://jqlang.github.io/jq/download/
# Step 1:
brew install jq

# Step 2:
Download log archive from S3

# Step 3:
mkdir cloudtrail && mv <filename-of-archive.json> cloudtrail/ && cd cloudtrail

# Search the file for user identities:
find . -type f -exec jq '.Records[].userIdentity.arn' {} \\;
# This will search all files in our directory, then run the `jq` command looking for `userIdentity.arn`.
Note that this is case sensitive, so if you capitalize the `User` for example, it won’t find anything.

# Example result:
"arn:aws:sts::212301213533:assumed-role/lab/christophe"

# Extract items from the Records array and display the eventTime, sourceIPAddress, userIdentity ARN and
eventName
```

```
find . -type f -exec jq -r '.Records[] | [.eventTime, .sourceIPAddress, .userIdentity.arn, .eventName] |
join(" -- ")' {} \\; | sort
```

Event History & Insights Events

```
# See a list of the latest events in JSON format
lookup-events --max-items 1 --output json

# Instead of specifying a max limit for items returned, we can specify a time range, like this:
lookup-events --start-time <timestamp> --end-time <timestamp>

# Example:
aws cloudtrail lookup-events --start-time "2023-11-15, 5:00PM" --end-time "2023-11-15, 6:00 PM" --output
json

# The timestamps take in UNIX time values, and you can specify the date, month, and year values separate
d by forward slashes or hyphens.
1422317782
1422317782.0
01-27-2015
01-27-2015,01:16PM
"01-27-2015, 01:16 PM"
"01/27/2015, 13:16"
2015-01-27
"2015-01-27, 01:16 PM"

# We can also query by using a single attribute, like this:
aws cloudtrail lookup-events --lookup-attributes AttributeKey=<attribute>,AttributeValue=<string>

# For the AttributeKey, you can use these values (and they are case sensitive):
#- AccessKeyId
#- EventId
#- EventName
#- EventSource
#- ReadOnly
#- ResourceName
#- ResourceType
#- Username

# Example scenario - let’s say that you think an access key has potentially been compromised and has bee
n used by a threat actor. You can query the last 10 events related to that access key like this:
aws cloudtrail lookup-events --lookup-attributes AttributeKey=AccessKeyId,AttributeValue=EXAMPLE_KEY_ID_
HERE --max-items 10

# Example scenario - if you believe that a role has been used for malicious purposes, you can search for
Events related to that role, like:
aws cloudtrail lookup-events --lookup-attributes AttributeKey=ResourceName,AttributeValue=AWSServiceRole
ForOrganizations --max-items 2

# Example scenario - as another similar example, you can search for events related to specific users by
using the Username AttributeKey:
aws cloudtrail lookup-events --lookup-attributes AttributeKey=Username,AttributeValue=christophe --max-i
tems 2
```

CloudTrail Lake

```
# Create and manage event data stores
create-event-data-store --name [--advanced-event-selectors <value>]

# Return info about a specific event data store
get-event-data-store --event-data-store <value> # The ARN (or ID suffix of the ARN)

# List all event data stores
# Returns an array of EventDataStores[]
list-event-data-stores

# Starts ingesting live events for a specific event data store
start-event-data-store-ingestion --event-data-store <value> # The ARN (or ID suffix of the ARN)

# Stop ingesting live events for a specific event data store (for cost management)
stop-event-data-store-ingestion --event-data-store <value> # The ARN (or ID suffix of the ARN)

# Create a channel for CloudTrail to ingest events from a partner or external source. After you create a
channel, a CloudTrail Lake event data store can log events from the partner or source that you specify.
create-channel --name <value> --source <value> --destinations <value> # Event data stores to send events
from this channel to

# Return a specific channel's info
get-channel --channel

# List all channels
list-channels

# Returns metadata about a query
aws cloudtrail describe-query --event-data-store ac1e7a13-7fae-45e6-aa26-aaefd2a44558 --query-id 95f234d
d-b848-4f9b-ac95-3b986cf24880 # [or --query-alias <value>]

# Get the results of a query run in Lake
get-query-results --query-id <value>

# Example:
aws cloudtrail get-query-results --query-id 95f234dd-b848-4f9b-ac95-3b986cf24880

#Remove: real ID 9bed2e1f-4a65-4f8c-bc0a-fe43a9733322

# List all queries run in the last 7 days
# Returns a Queries[] array
list-queries --event-data-store <value>

# Example:
aws cloudtrail list-queries --event-data-store 2cbe7a73-7fae-45e6-aa26-aaefd2a44558

# Imports data from logged trail events in S3 to an event data store for Lake queries
start-import [--destinations <value>] [--import-source <value>] [--start-event-time <value>] [--end-even
t-time <value>]

stop-import --import-id <value>

# Run a CloudTrail Lake query
# --query-statement is the SQL code to run
```

```
# --query-parameters is the list of query parameters
start-query [--query-statement <value>] [--delivery-s3-uri <value>] [--query-parameters <value>]
```

Insights Events

```
# Enables Insights event logging for an existing trail or event data store
# --insight-selectors would be the Insights types you want to log (ApiCallRateInsight and/or ApiErrorRat
eInsight)
put-insight-selectors [--trail-name <value>] --insight-selectors <value> [--event-data-store <value>] [-
-insights-destination <value>]

# Retrieve settings for Insights event selectors configured for trails or event data stores
get-insight-selectors [--trail-name <value>][--event-data-store <value>]
```

Next steps

Congratulations! You've just completed our Beginner's Guide to CloudTrail for Security course!

This course was meant to give you a solid foundation and understanding of how CloudTrail works and how you can configure it for your use case and your organization.

As a next step, I recommend learning how to use CloudTrail for Threat Detection and Response. While we are going to release a course that teaches how to do this, in the meantime, practice creating potential scenarios and find ways of using all the features you've just learned about to detect the threat and gather evidence as if you were doing it on the job.

If you need inspiration, check out our exploitation courses at Cybr for ideas on various types of attacks you could mimic in your account.

Thank you so much for enrolling in my course and I hope you enjoyed it!

Outro & thank you!

We appreciate you purchasing our Ebook and supporting Cybr! As always, if you have any questions or need any assistance, you can find us on Discord or email me directly at christophe@cybr.com

Copyright © 2023 by Cybr, Inc. All Rights Reserved.