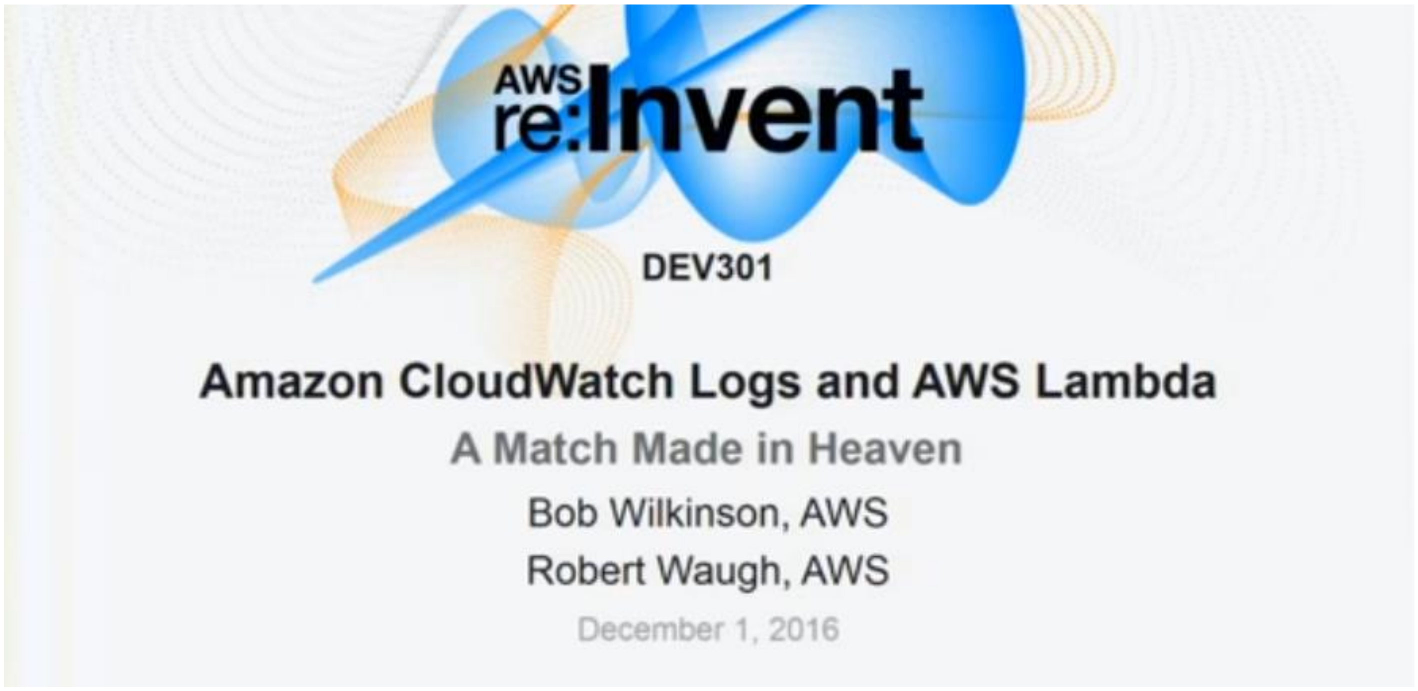


AWS re:Invent

DEV301

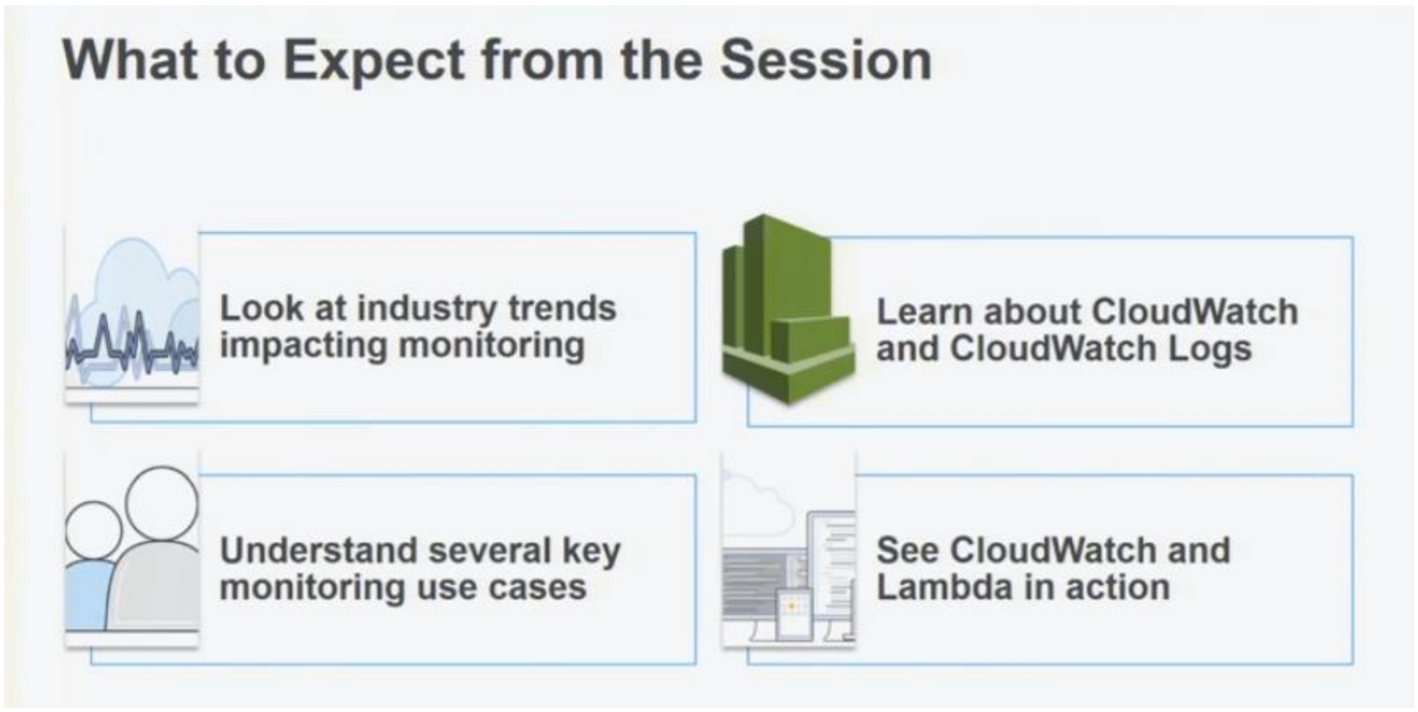
Amazon CloudWatch Logs and AWS Lambda

A Match Made in Heaven

Bob Wilkinson, AWS

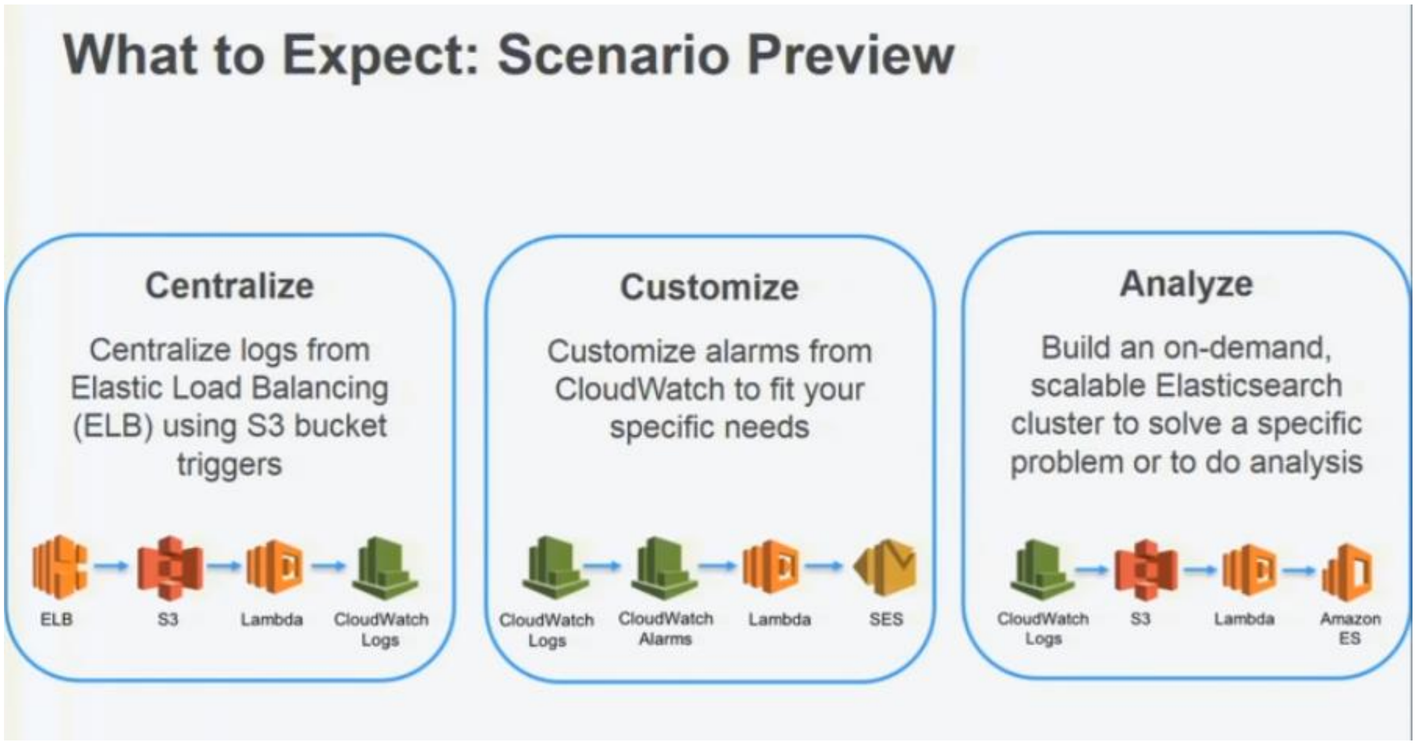
Robert Waugh, AWS

December 1, 2016



What to Expect from the Session

- Look at industry trends impacting monitoring
- Learn about CloudWatch and CloudWatch Logs
- Understand several key monitoring use cases
- See CloudWatch and Lambda in action



What to Expect: Scenario Preview

Centralize

Centralize logs from Elastic Load Balancing (ELB) using S3 bucket triggers

Customize

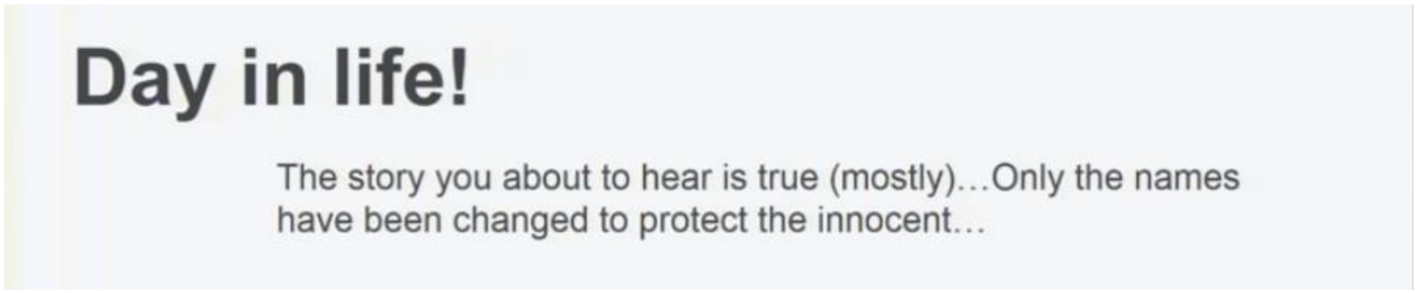
Customize alarms from CloudWatch to fit your specific needs

Analyze

Build an on-demand, scalable Elasticsearch cluster to solve a specific problem or to do analysis

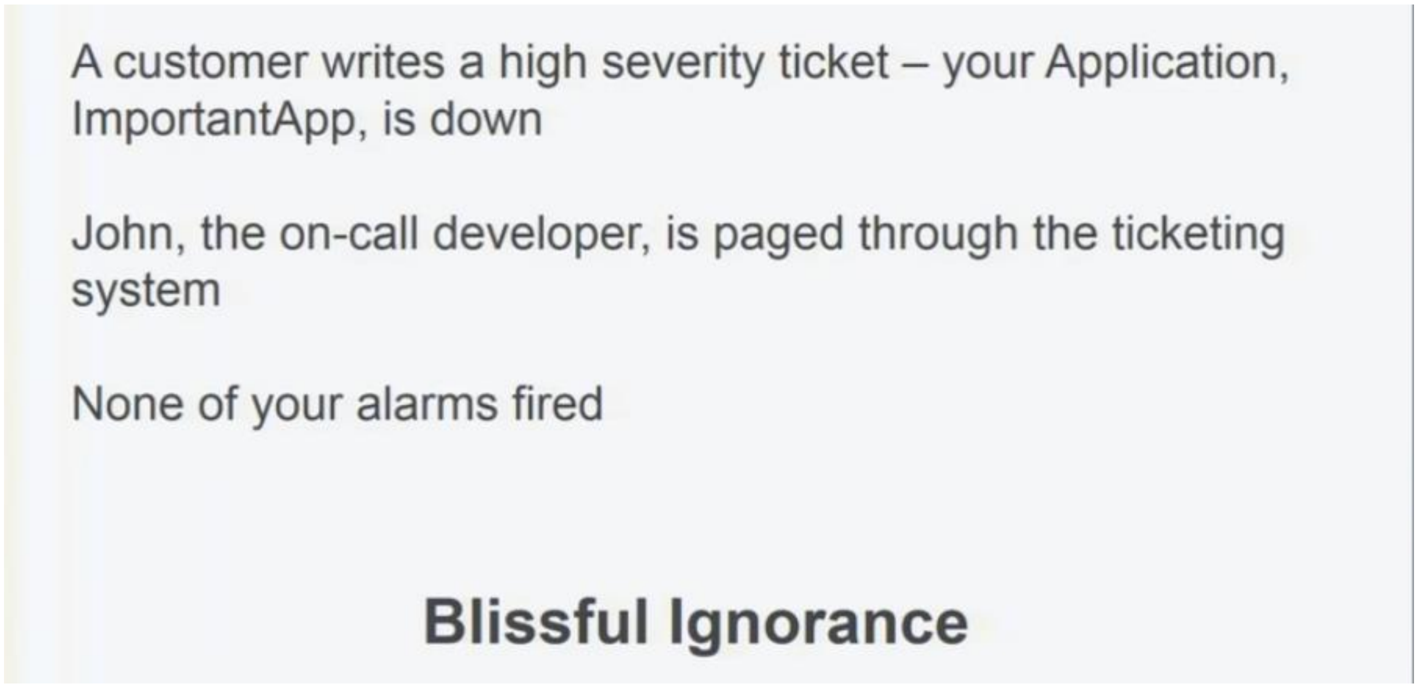
Recognize This?

This dog has a monitoring problem



Day in life!

The story you about to hear is true (mostly)…Only the names have been changed to protect the innocent…

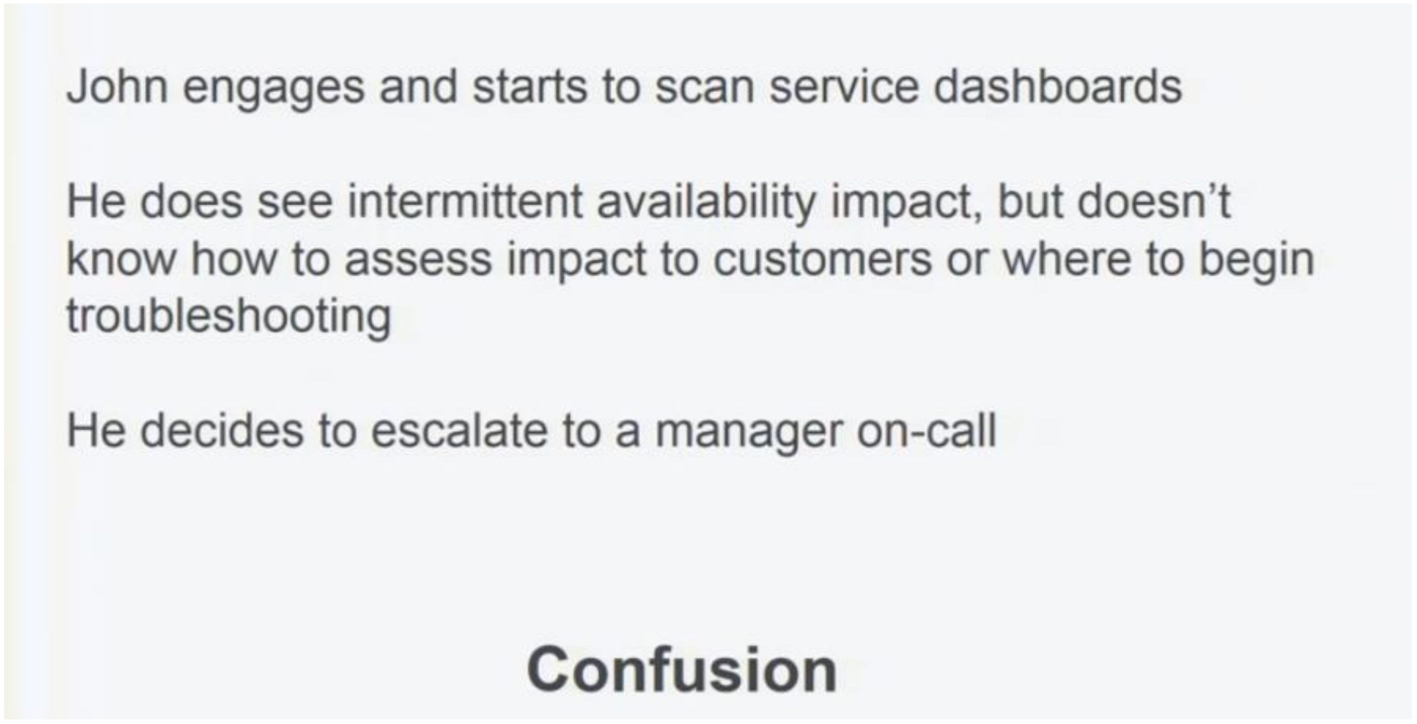


A customer writes a high severity ticket – your Application, ImportantApp, is down

John, the on-call developer, is paged through the ticketing system

None of your alarms fired

Blissful Ignorance

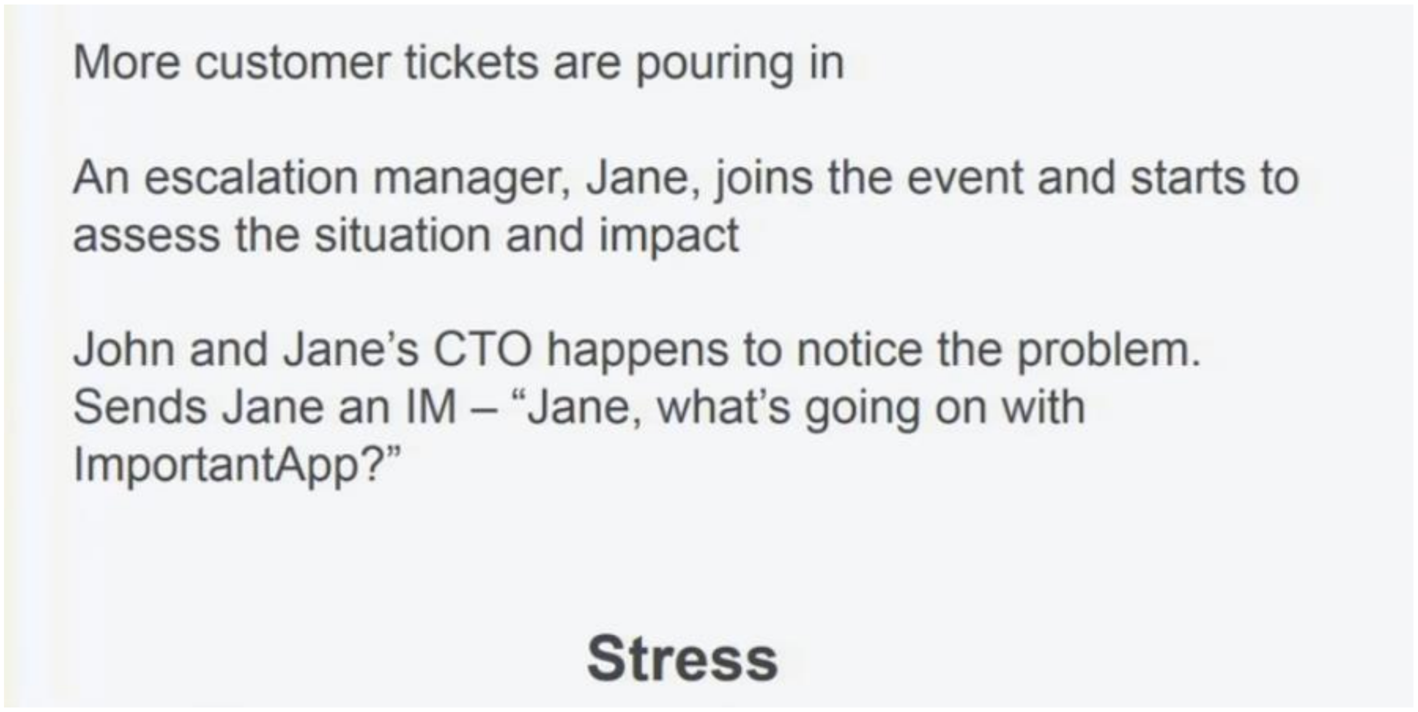


John engages and starts to scan service dashboards

He does see intermittent availability impact, but doesn't know how to assess impact to customers or where to begin troubleshooting

He decides to escalate to a manager on-call

Confusion

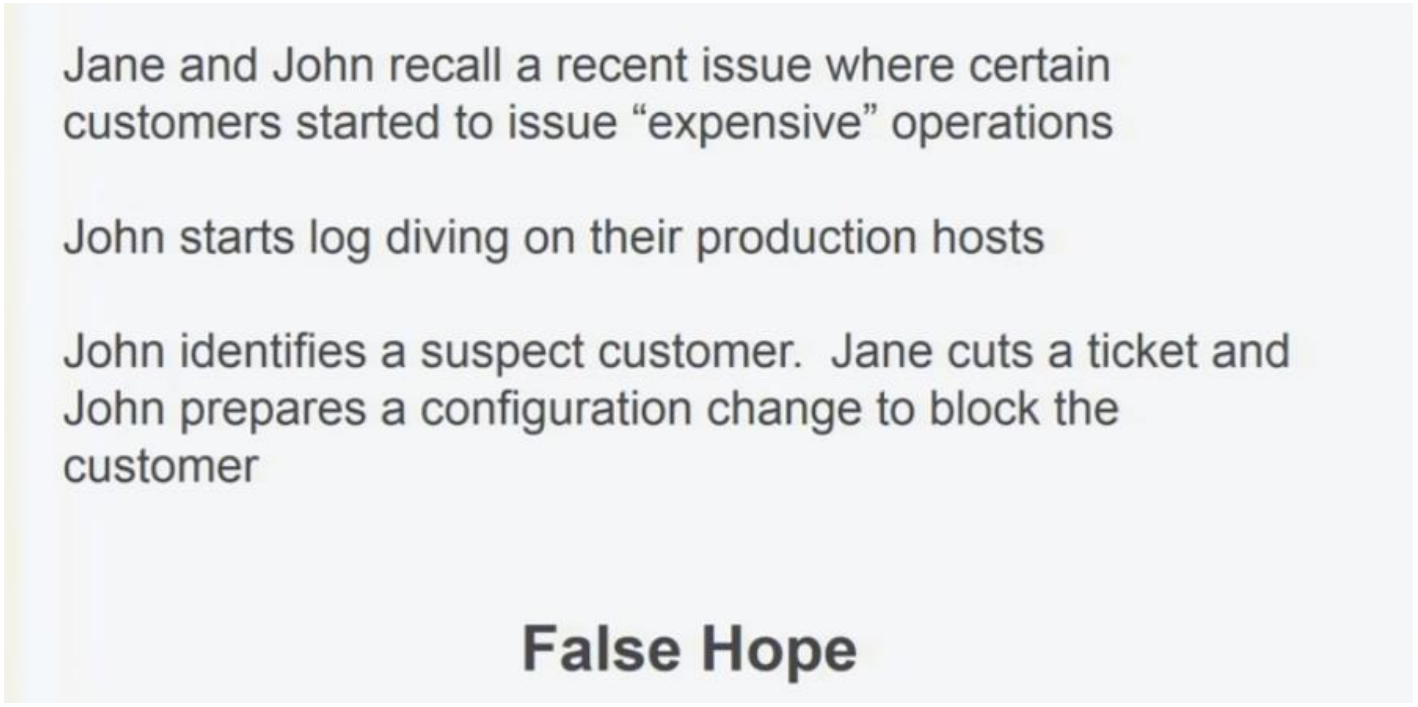


More customer tickets are pouring in

An escalation manager, Jane, joins the event and starts to assess the situation and impact

John and Jane’s CTO happens to notice the problem. Sends Jane an IM – “Jane, what’s going on with ImportantApp?”

Stress



Jane and John recall a recent issue where certain customers started to issue “expensive” operations

John starts log diving on their production hosts

John identifies a suspect customer. Jane cuts a ticket and John prepares a configuration change to block the customer

False Hope

The other team engages and indicates they didn't change anything

Jane and John also confirm this when the availability impact persists after deploying the configuration change

Out of ideas, John suggests to fail over to the standby – "It can't hurt…"

Desperation

After the failover, ImportantApp recovers (Yay!)

Our root cause deep dive finds that a new JDBC version introduced a memory leak leading to Java heap exhaustion

We fix the leak, add new alarms on memory usage, and tune our service alarms

Enlightenment

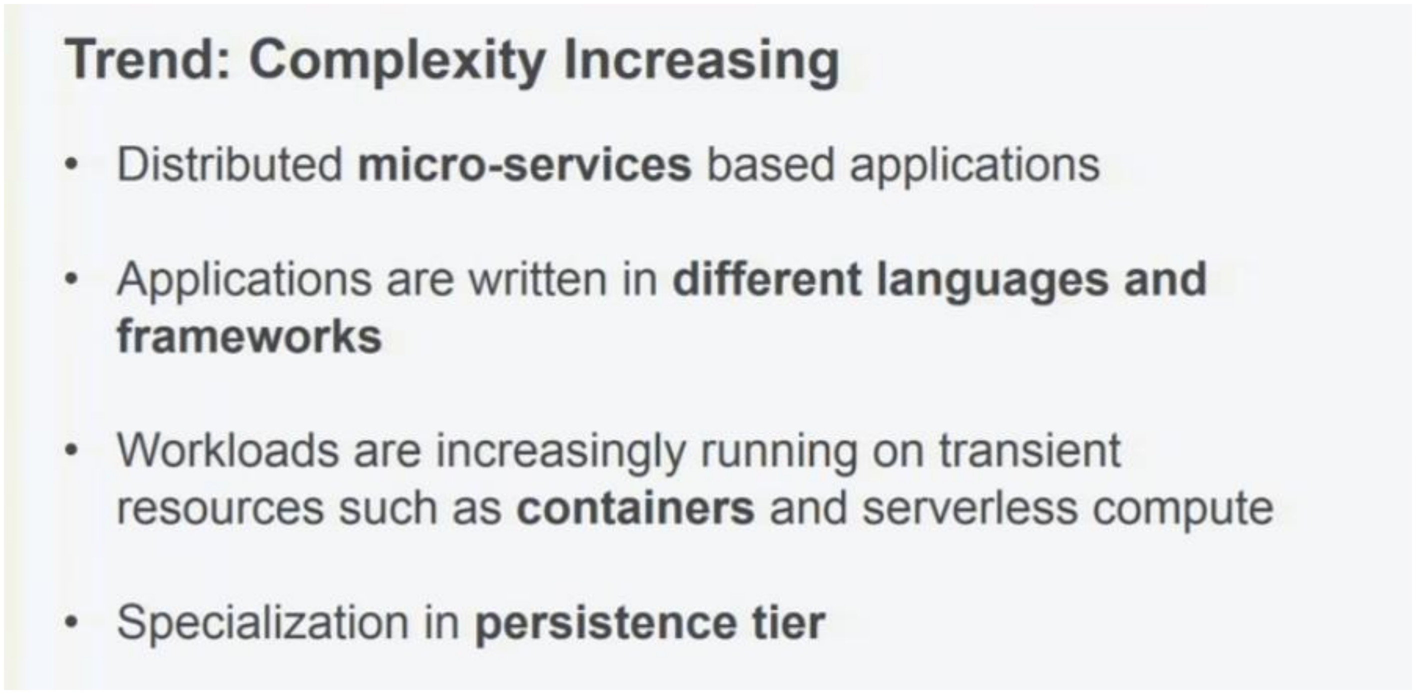
Can we do better than that?

Day in the Life - Reflection

- We have missing alarms and some of the alarms we have are not actionable
- We do not always have the right logs and interacting with them can be tedious
- Our dashboards do not tell us enough about customer impact or behavior changes

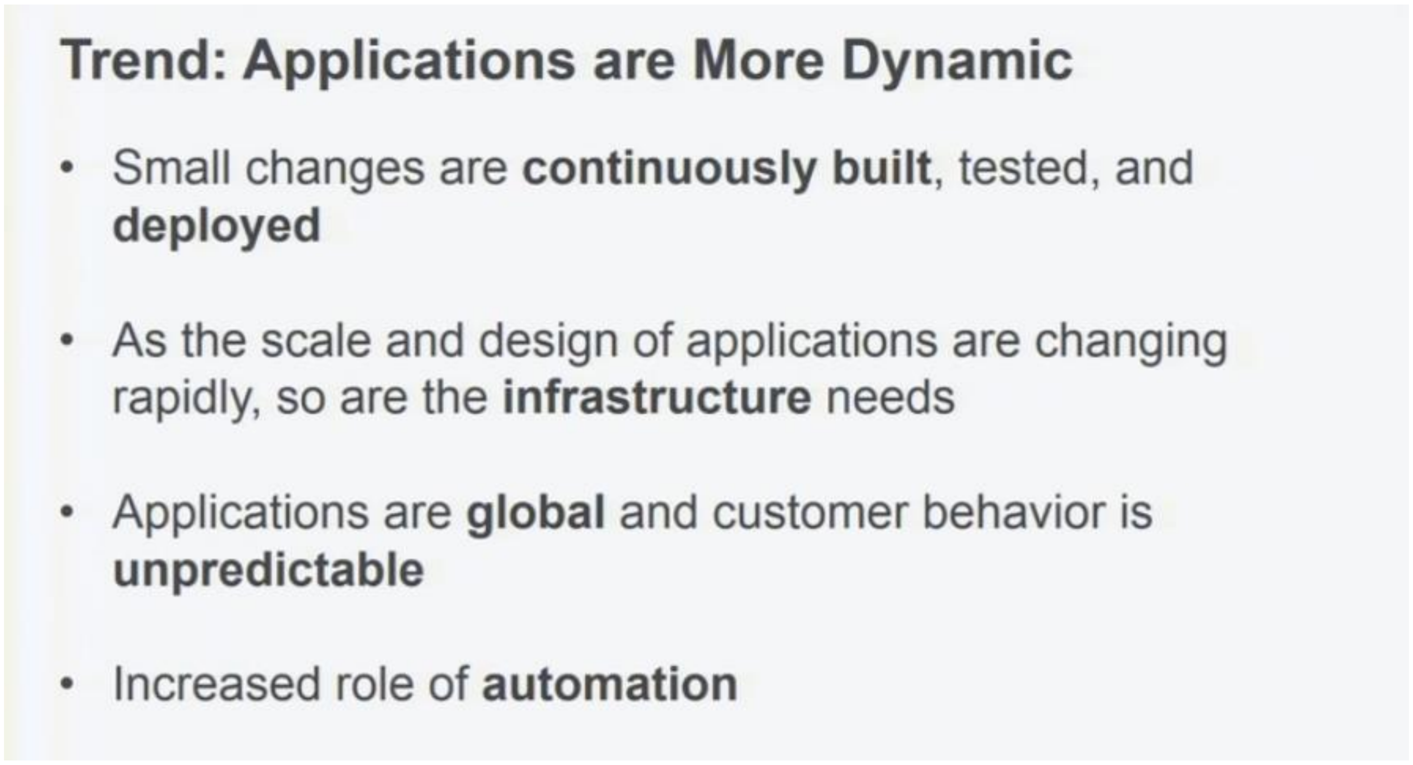
Monitoring is really (really) hard!

Trends in Monitoring



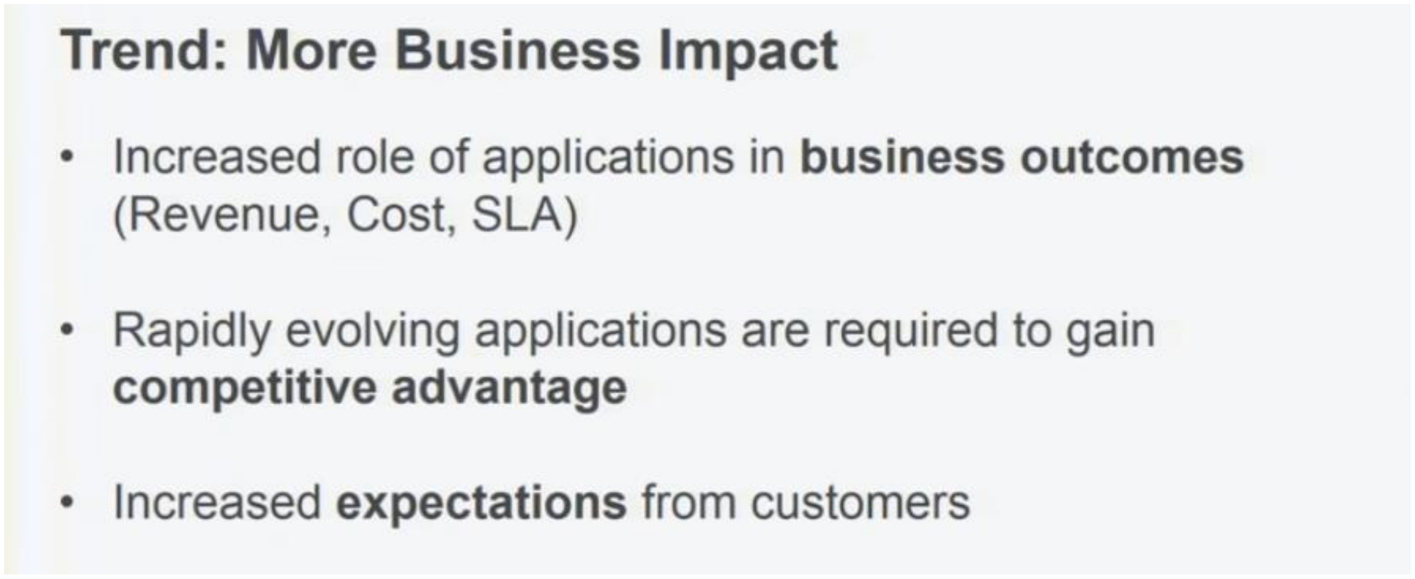
Trend: Complexity Increasing

- Distributed **micro-services** based applications
- Applications are written in **different languages and frameworks**
- Workloads are increasingly running on transient resources such as **containers** and serverless compute
- Specialization in **persistence tier**



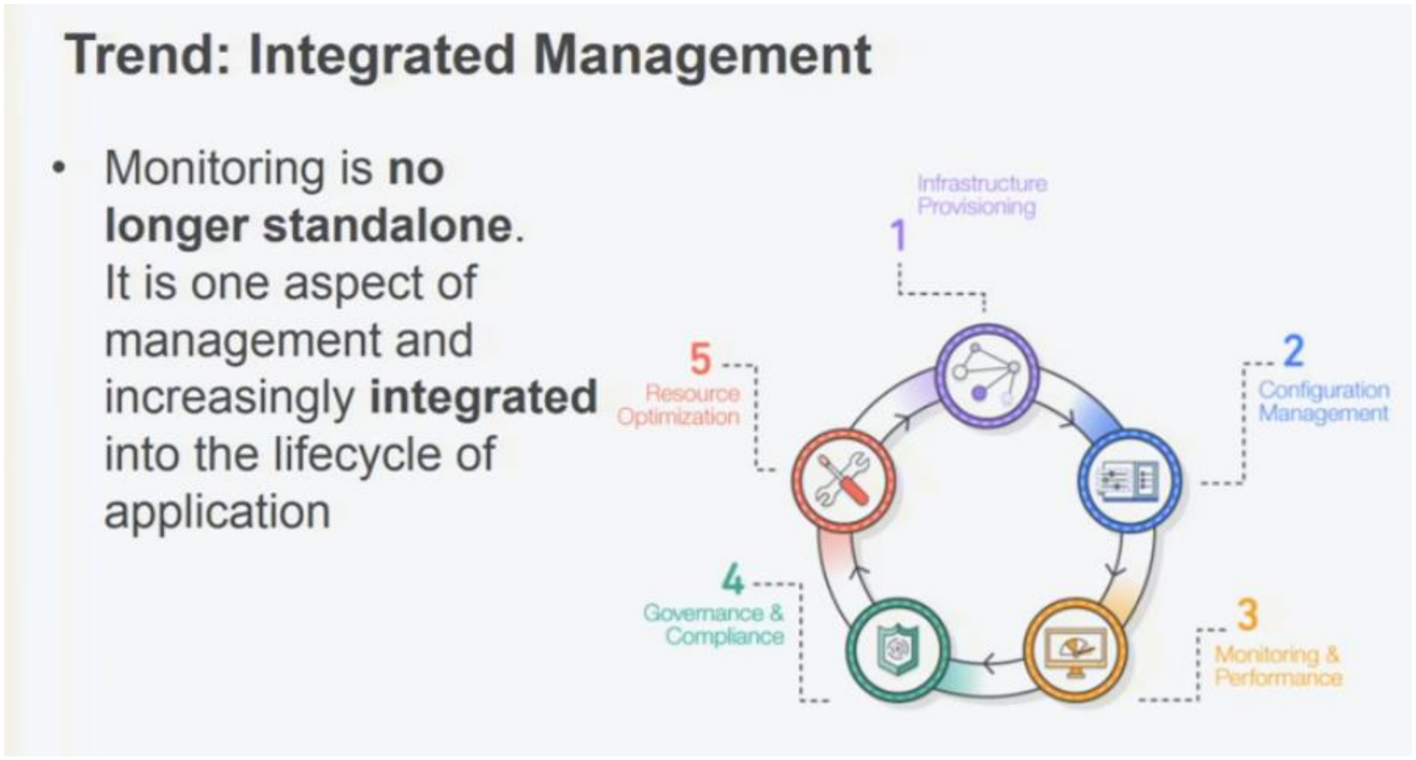
Trend: Applications are More Dynamic

- Small changes are **continuously built**, tested, and **deployed**
- As the scale and design of applications are changing rapidly, so are the **infrastructure** needs
- Applications are **global** and customer behavior is **unpredictable**
- Increased role of **automation**



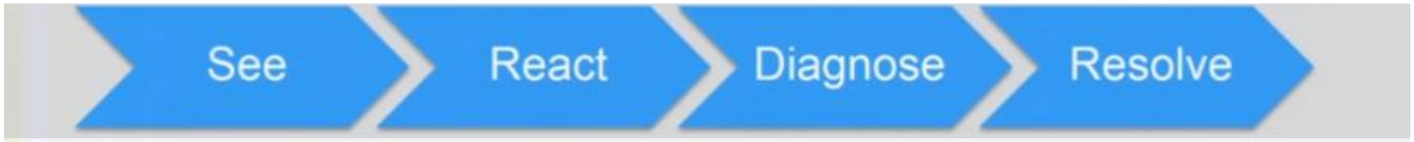
Trend: More Business Impact

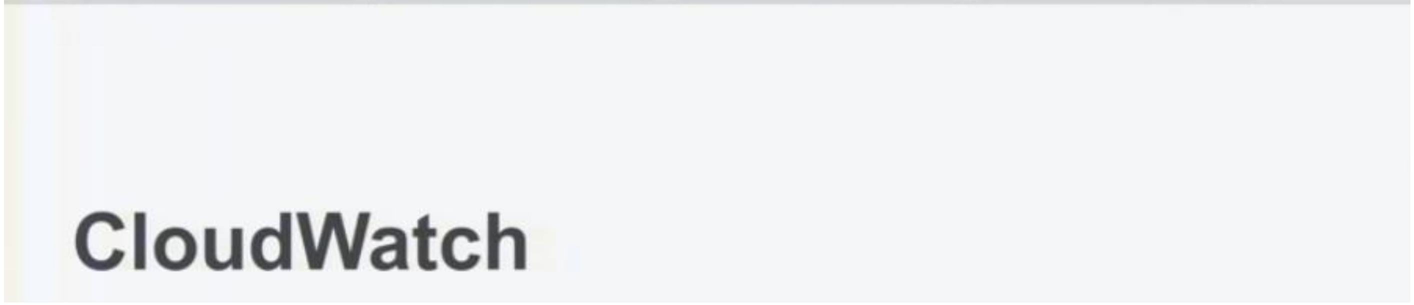
- Increased role of applications in **business outcomes** (Revenue, Cost, SLA)
- Rapidly evolving applications are required to gain **competitive advantage**
- Increased **expectations** from customers



Trend: Integrated Management

- Monitoring is **no longer standalone**. It is one aspect of management and increasingly **integrated** into the lifecycle of application





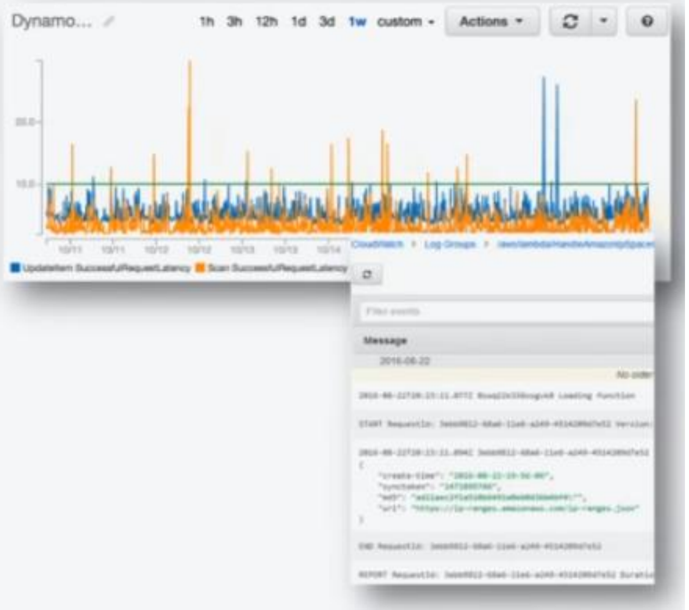
CloudWatch

CloudWatch ***(CF)*** Is a portfolio of tools covering metrics, alarms, events, dashboards, etc.

See | React | Diagnose | Resolve

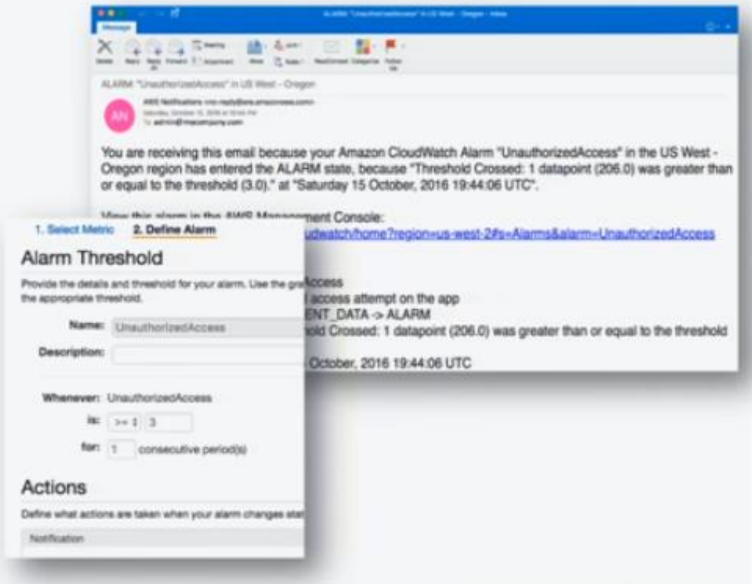
Use AWS generated metrics, logs, and events over time to understand the behavior of your system

Publish custom metrics, logs, and events for your application specific telemetry


Dynamo...
1h 3h 12h 1d 3d 1w custom
Actions
UpdateItem SuccessfulRequestLatency
Scan SuccessfulRequestLatency
CloudWatch > Log Groups
Message

See | React | Diagnose | Resolve

Trigger automatic notifications based on your own rules and metric thresholds


ALARM: "UnauthorizedAccess" in US West - Oregon

You are receiving this email because your Amazon CloudWatch Alarm "UnauthorizedAccess" in the US West - Oregon region has entered the ALARM state, because "Threshold Crossed: 1 datapoint (206.0) was greater than or equal to the threshold (3.0)." at "Saturday 15 October, 2016 19:44:06 UTC".

View this alarm in the AWS Management Console:
udwatch/home?region=us-west-2#s=Alarms&alarm=UnauthorizedAccess

Access
access attempt on the app
ENT_DATA -> ALARM
old Crossed: 1 datapoint (206.0) was greater than or equal to the threshold
October, 2016 19:44:06 UTC

1. Select Metric 2. Define Alarm

Alarm Threshold

Provide the details and threshold for your alarm. Use the gr
the appropriate threshold.

Name: UnauthorizedAccess
Description:

Whenever: UnauthorizedAccess
is: >= 3
for: 1 consecutive period(s)

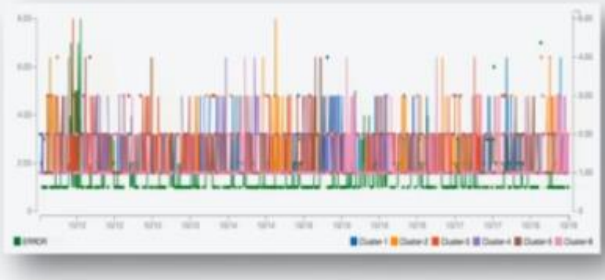
Actions

Define what actions are taken when your alarm changes sta

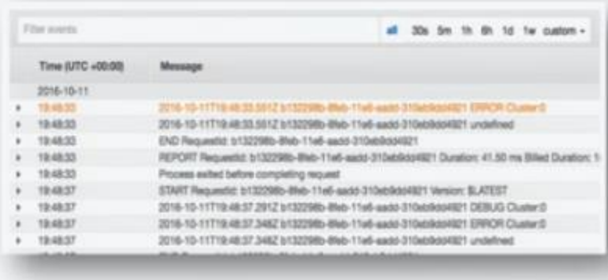
Notification

See React **Diagnose** Resolve

Inspect, navigate, zoom, and correlate across time to investigate issues



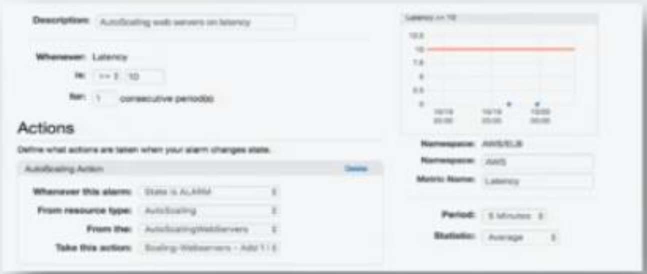
Jump to your logs directly from your metrics to perform searches or **generate additional metrics from log data**



```
Time (UTC +00:00)   Message
2016-10-11
19:48:33   2016-10-11T19:48:33.551Z b13229bb-8feb-11e6-aadd-310eb9dd4921 ERROR Cluster:0
19:48:33   2016-10-11T19:48:33.551Z b13229bb-8feb-11e6-aadd-310eb9dd4921 undefined
19:48:33   END RequestId: b13229bb-8feb-11e6-aadd-310eb9dd4921
19:48:33   REPORT RequestId: b13229bb-8feb-11e6-aadd-310eb9dd4921 Duration: 41.50 ms Billed Duration: 1
19:48:33   Process exited before completing request
19:48:37   START RequestId: b13229bb-8feb-11e6-aadd-310eb9dd4921 Version: $LATEST
19:48:37   2016-10-11T19:48:37.291Z b13229bb-8feb-11e6-aadd-310eb9dd4921 DEBUG Cluster:0
19:48:37   2016-10-11T19:48:37.348Z b13229bb-8feb-11e6-aadd-310eb9dd4921 ERROR Cluster:0
19:48:37   2016-10-11T19:48:37.348Z b13229bb-8feb-11e6-aadd-310eb9dd4921 undefined
```

See React Diagnose **Resolve**

Easily and automatically correct issues via common actions that you control



Description: AutoScaling web servers on latency

Whenever: Latency
is: >= 10
for: 1 consecutive period(s)

Actions

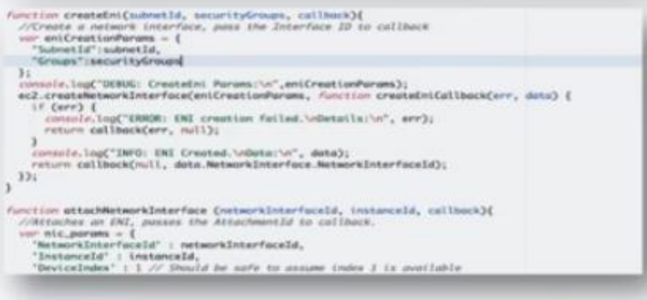
Define what actions are taken when your alarm changes state.

AutoScaling Action — Delete

Whenever this alarm: State is ALARM
From resource type: AutoScaling
From the: AutoScalingWebServers
Take this action: Scaling-Webservers - Add 1

Namespace: AWS/ELB
Namespace: AWS
Metric Name: Latency
Period: 5 Minutes
Statistic: Average

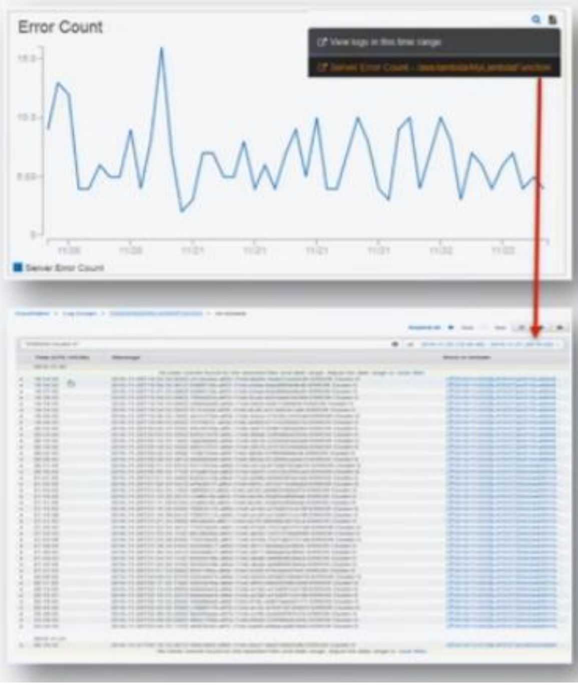
Define your own custom actions based on Lambda functions for more fine-grained control



```
function createEni(subnetId, securityGroups, callback){
  //Create a network interface, pass the Interface ID to callback
  var eniCreationParams = {
    "SubnetId":subnetId,
    "Groups":securityGroups
  };
  console.log("DEBUG: CreateEni Params:\n",eniCreationParams);
  ec2.createNetworkInterface(eniCreationParams, function createEniCallback(err, data) {
    if (err) {
      console.log("ERROR: ENI creation failed.\nDetails:\n", err);
      return callback(err, null);
    }
    console.log("INFO: ENI Created.\nData:\n", data);
    return callback(null, data.NetworkInterface.NetworkInterfaceId);
  });
}

function attachNetworkInterface (networkInterfaceId, instanceId, callback){
  //Attaches an ENI, passes the AttachmentId to callback.
  var nic_params = {
    'NetworkInterfaceId' : networkInterfaceId,
    'InstanceId' : instanceId,
    'DeviceIndex' : 1 // Should be safe to assume index 1 is available
```

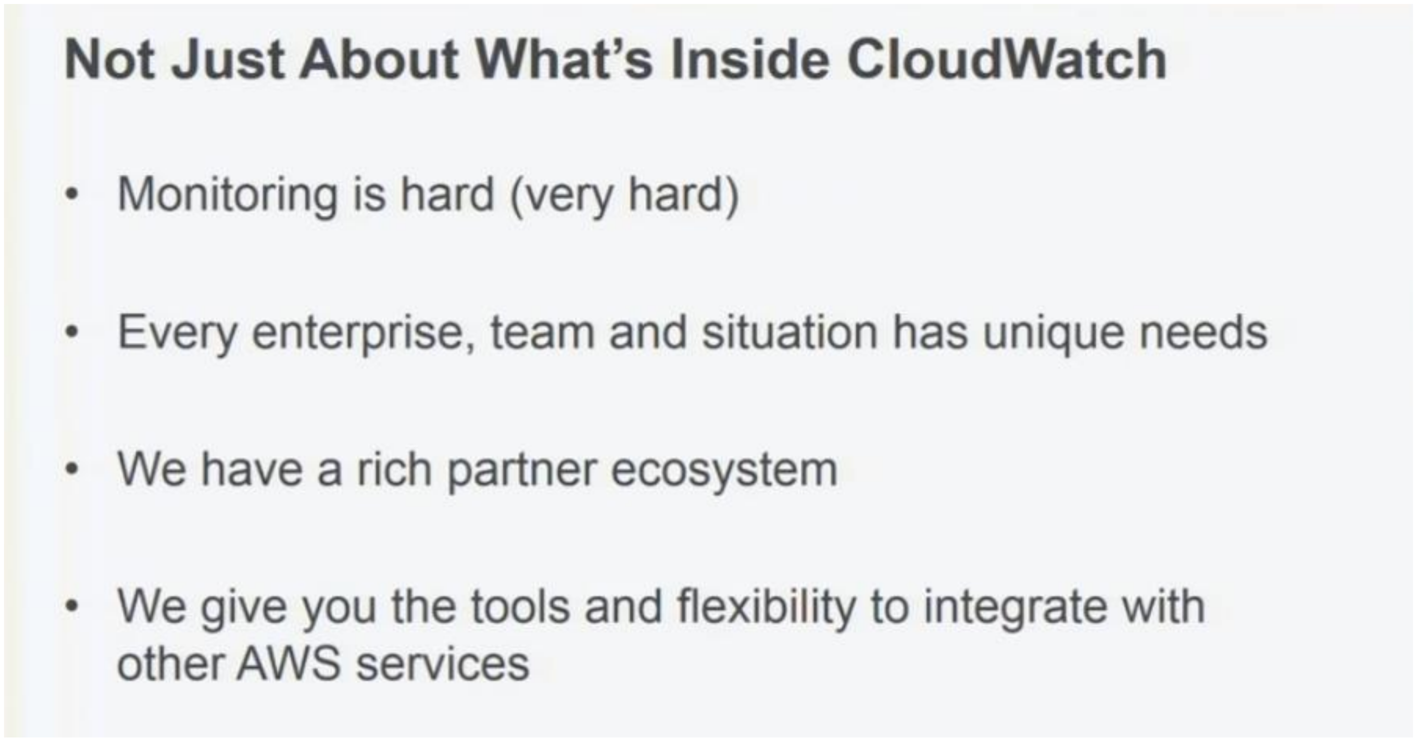
Recent Improvements

- Metrics Price Drop
- More metrics, logs, events from AWS services: CloudTrail, Elastic Beanstalk, SES
- Simple navigation from Metrics to your Logs
- Upgraded metric retention from 2 weeks to up to 15 months



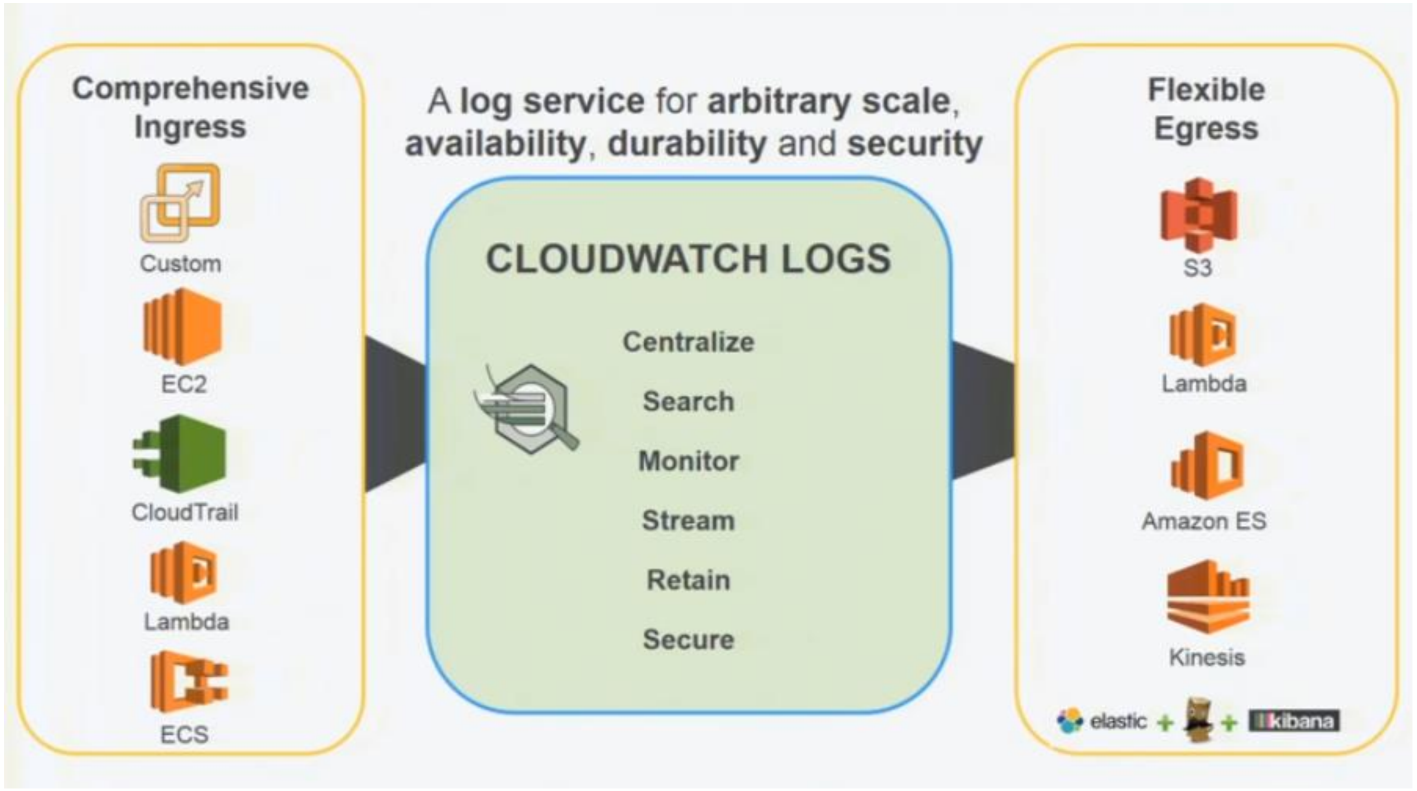
Recent Improvements (continued)

- Support for arbitrary metric percentiles
- collectd output plugin to simplify metric collection
- Improvements in Dashboards (new widgets, dark theme, Y axis limits)
- Improved Logs console experience



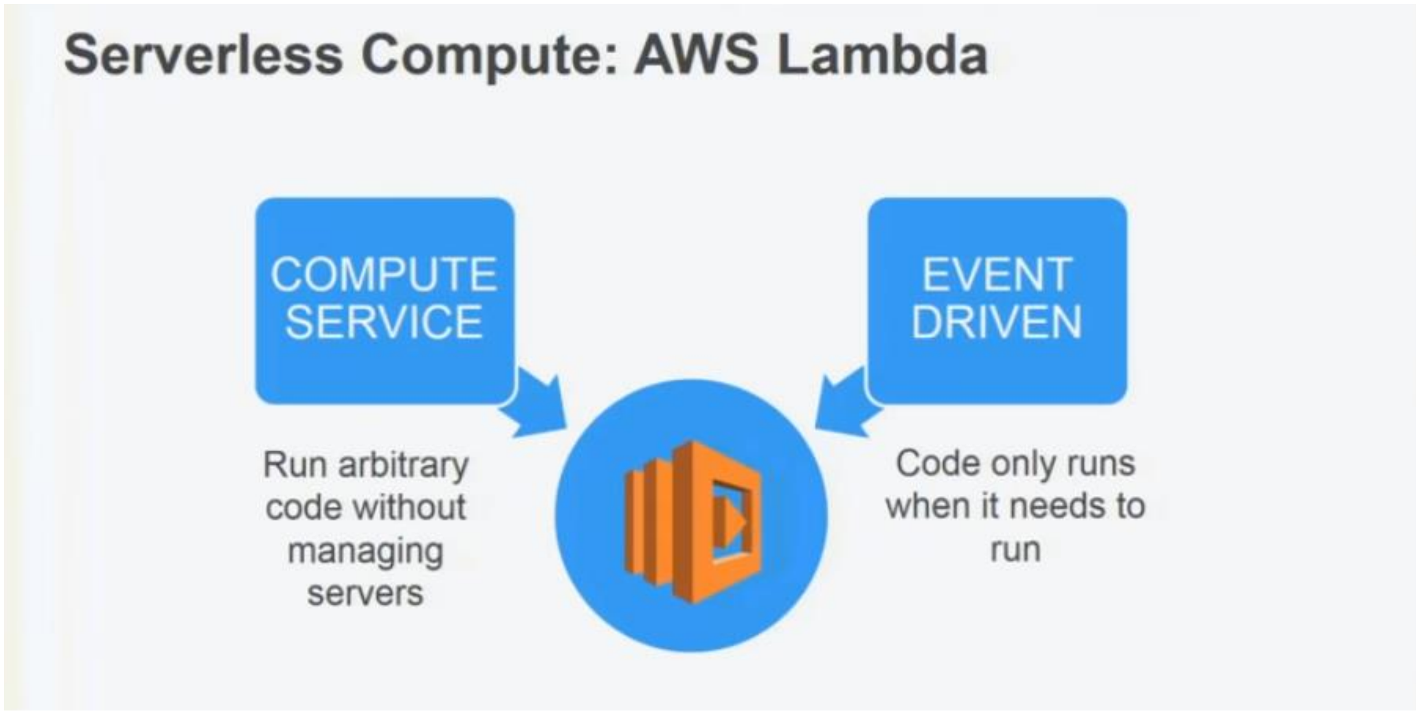
Not Just About What's Inside CloudWatch

- Monitoring is hard (very hard)
- Every enterprise, team and situation has unique needs
- We have a rich partner ecosystem
- We give you the tools and flexibility to integrate with other AWS services



For the ingress option, we have a log agent that you can install on any host that allows you to point to a log file to ship logs from within your host and stream the logs out in near real-time. You can then run text search on your logs and also extract metrics from your logs. You can take a data element within your logs and have it emitted out as a metric that you can put on a graph, set an alarm on, etc.

On the egress side of logs, you can use your filtered logs to egress things to a Kinesis stream, Elasticsearch, pipe it through a lambda function, do batch export of your logs to an S3 bucket, etc

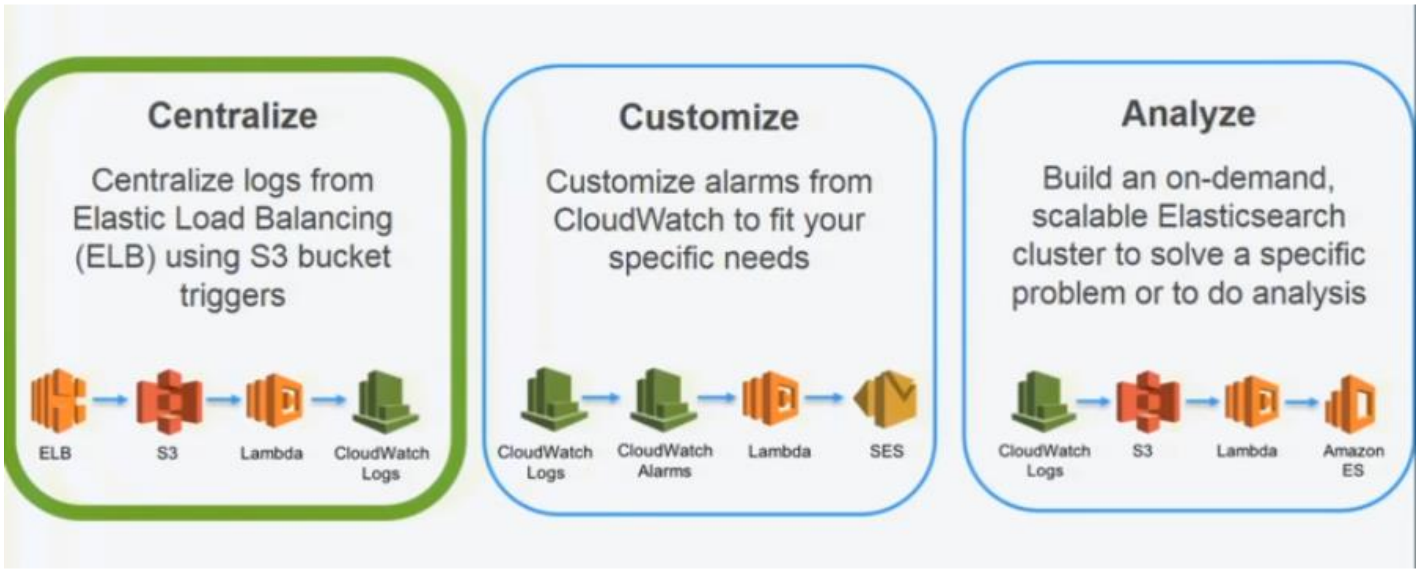

Serverless Compute: AWS Lambda

COMPUTE SERVICE

Run arbitrary code without managing servers

EVENT DRIVEN

Code only runs when it needs to run


Centralize

Centralize logs from Elastic Load Balancing (ELB) using S3 bucket triggers

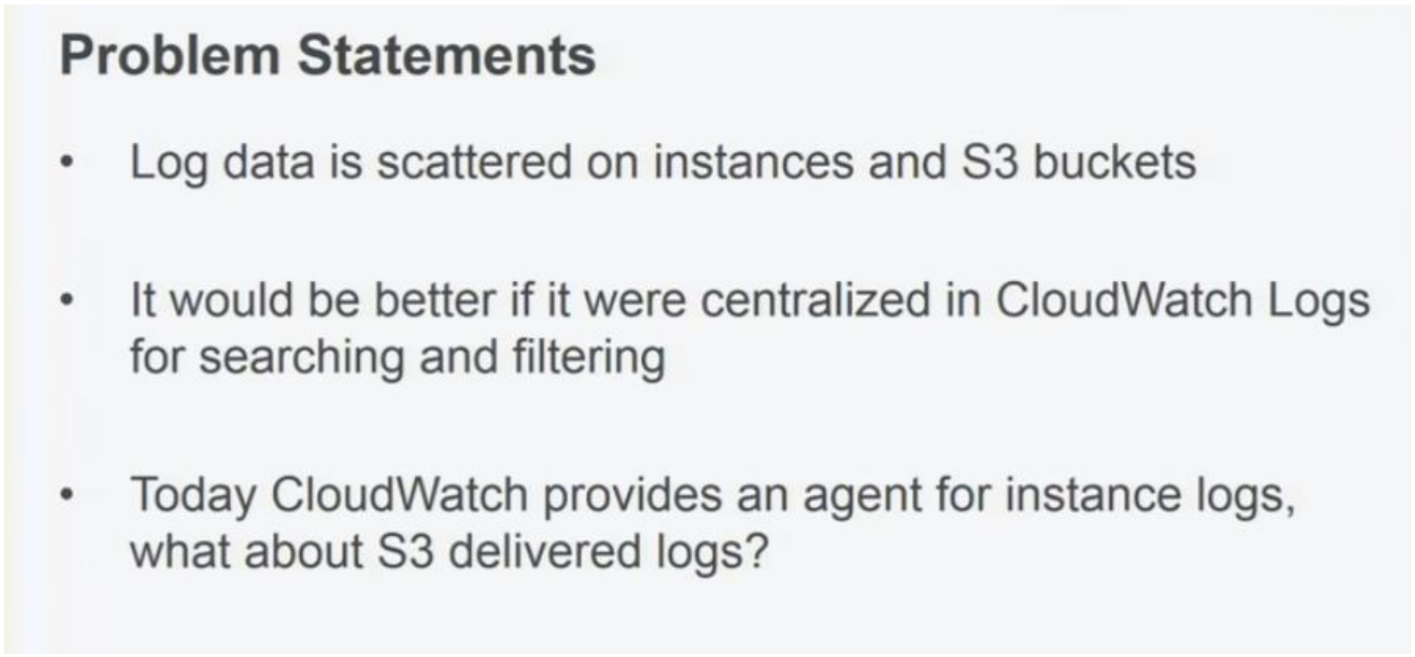
Customize

Customize alarms from CloudWatch to fit your specific needs

Analyze

Build an on-demand, scalable Elasticsearch cluster to solve a specific problem or to do analysis

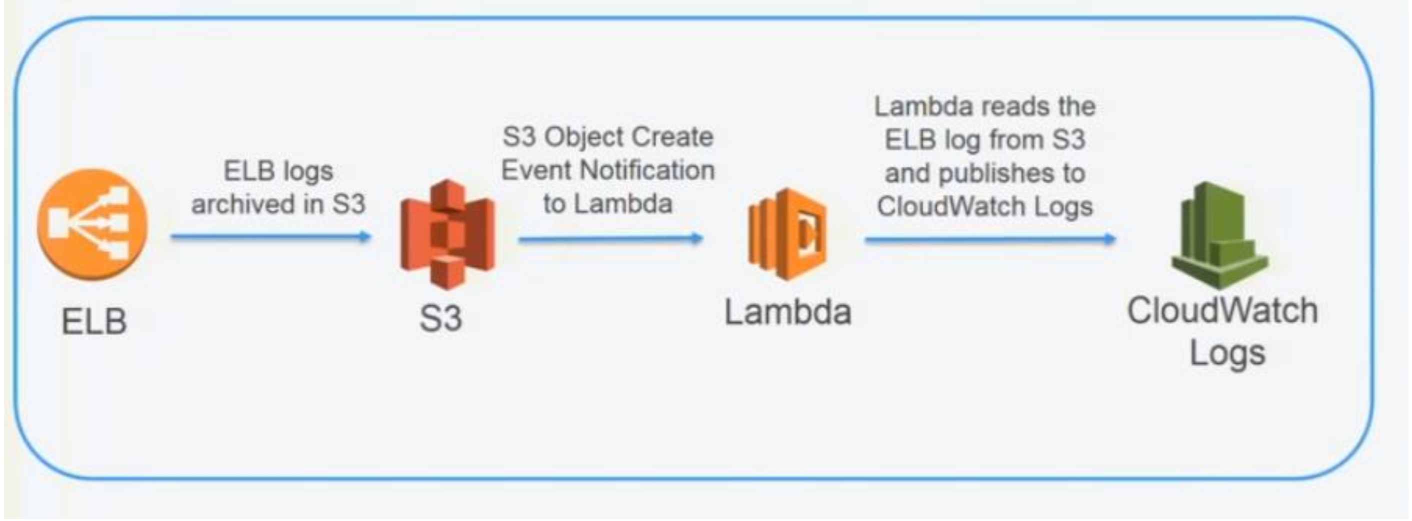
Having centralized logs from our instances and buckets in a single repository location with a consistent APIs, retention policy and access controls. You need to think about how to federate all that data into a single place.



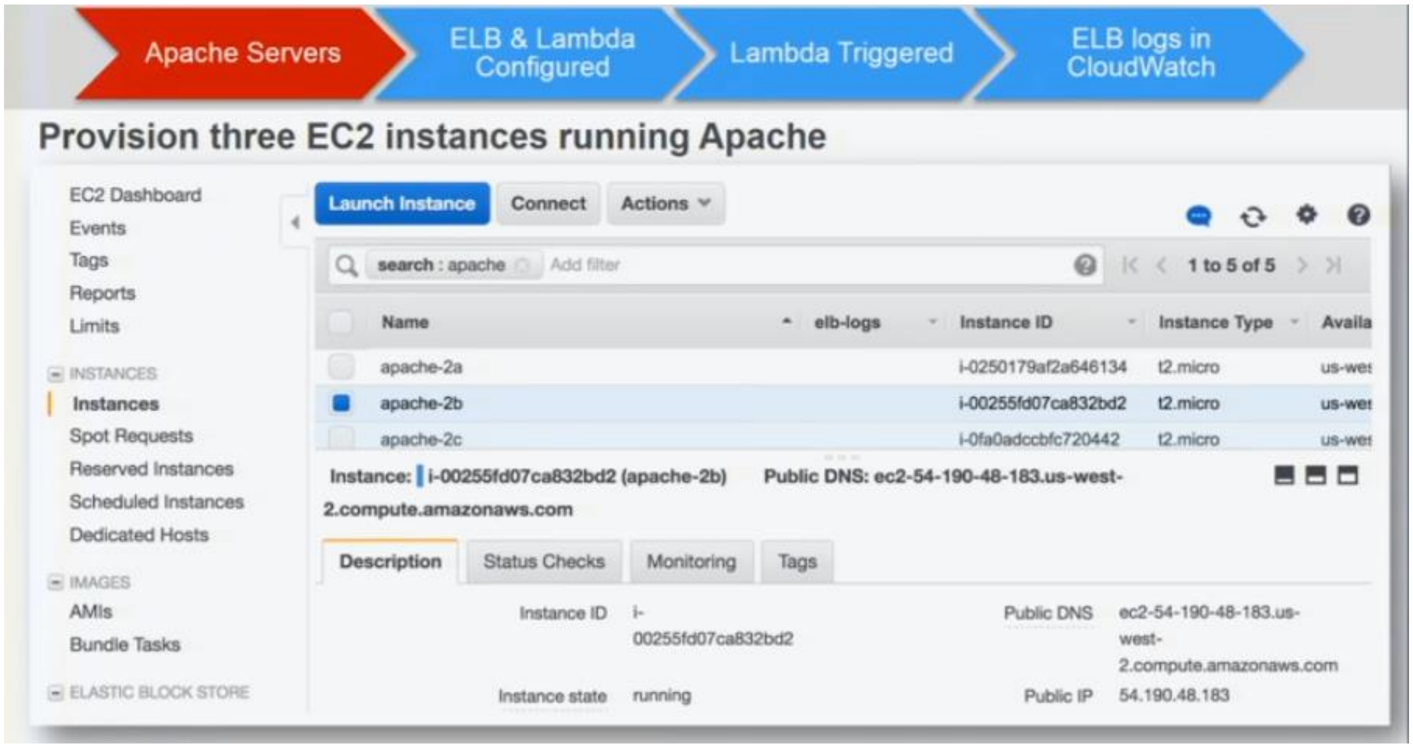
Problem Statements

- Log data is scattered on instances and S3 buckets
- It would be better if it were centralized in CloudWatch Logs for searching and filtering
- Today CloudWatch provides an agent for instance logs, what about S3 delivered logs?

Flow of Events


ELB
ELB logs archived in S3
S3
S3 Object Create Event Notification to Lambda
Lambda
Lambda reads the ELB log from S3 and publishes to CloudWatch Logs
CloudWatch Logs

You can configure ELB to send your access logs to an S3 bucket, we then set up an object creation notification event to lambda that will trigger a lambda function to read and publish the logs to the central location



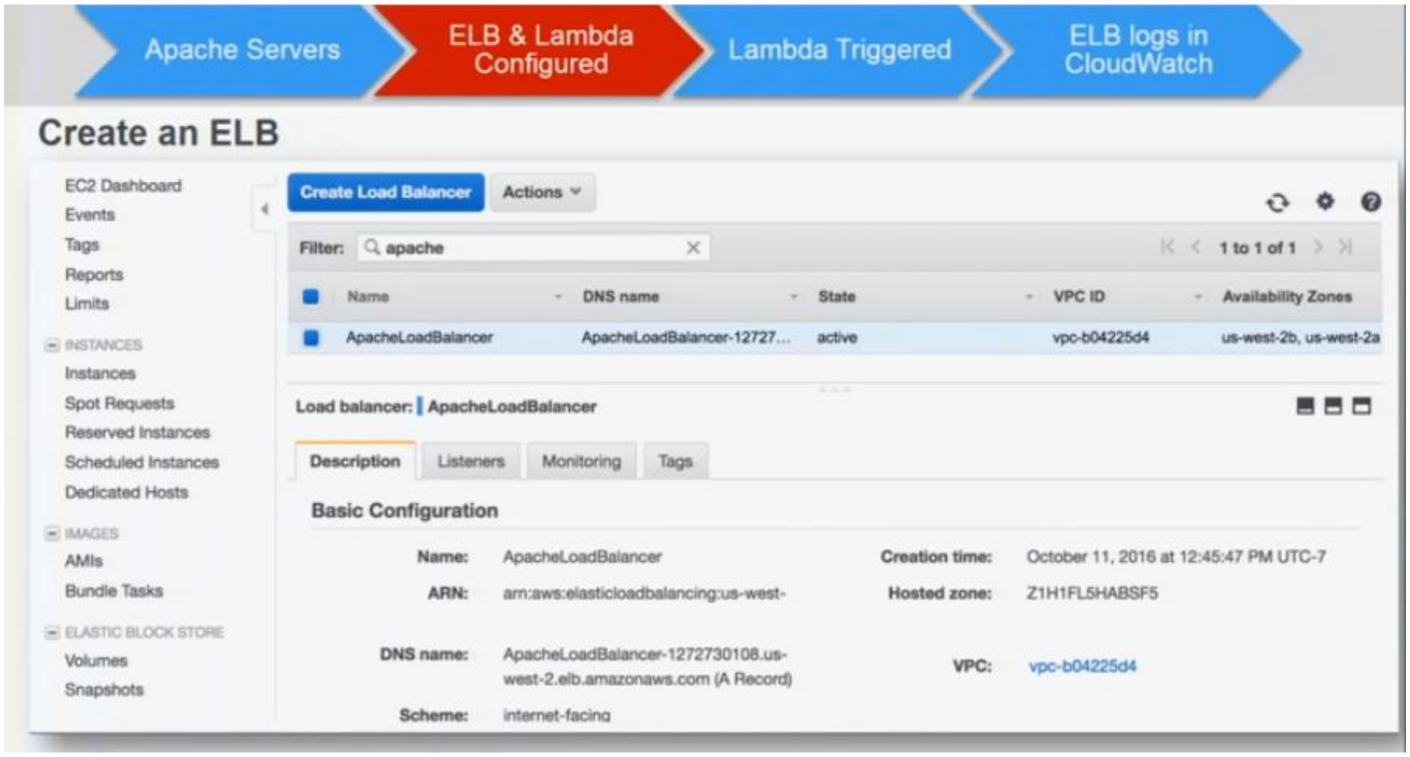
Provision three EC2 instances running Apache

search : apache   Add filter   1 to 5 of 5

Name	elb-logs	Instance ID	Instance Type	Availa
apache-2a		i-0250179af2a646134	t2.micro	us-wes
apache-2b		i-00255fd07ca832bd2	t2.micro	us-wes
apache-2c		i-0fa0adccbfc720442	t2.micro	us-wes

Instance: i-00255fd07ca832bd2 (apache-2b)   Public DNS: ec2-54-190-48-183.us-west-2.compute.amazonaws.com

Instance ID: i-00255fd07ca832bd2
Instance state: running
Public DNS: ec2-54-190-48-183.us-west-2.compute.amazonaws.com
Public IP: 54.190.48.183



Create an ELB

Filter: apache   1 to 1 of 1

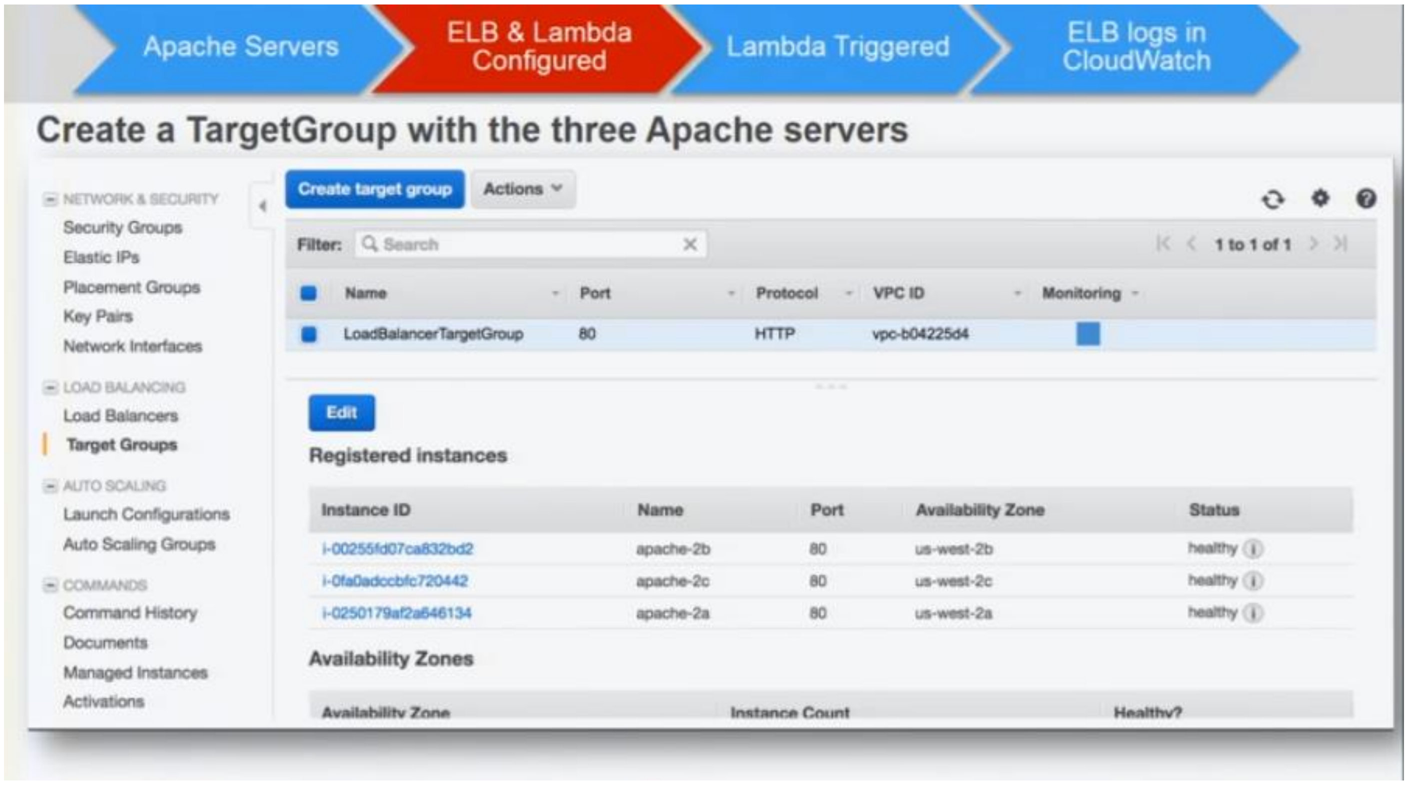
Name	DNS name	State	VPC ID	Availability Zones
ApacheLoadBalancer	ApacheLoadBalancer-12727...	active	vpc-b04225d4	us-west-2b, us-west-2a

Load balancer: ApacheLoadBalancer

Basic Configuration

Name: ApacheLoadBalancer
ARN: arn:aws:elasticloadbalancing:us-west-
DNS name: ApacheLoadBalancer-1272730108.us-west-2.elb.amazonaws.com (A Record)
Scheme: internet-facing
Creation time: October 11, 2016 at 12:45:47 PM UTC-7
Hosted zone: Z1H1FL5HABSF5
VPC: vpc-b04225d4

We then set up an ELB to front those 3 apache servers in the 3 different AZs



Create a TargetGroup with the three Apache servers

Name	Port	Protocol	VPC ID	Monitoring
LoadBalancerTargetGroup	80	HTTP	vpc-b04225d4	🟦

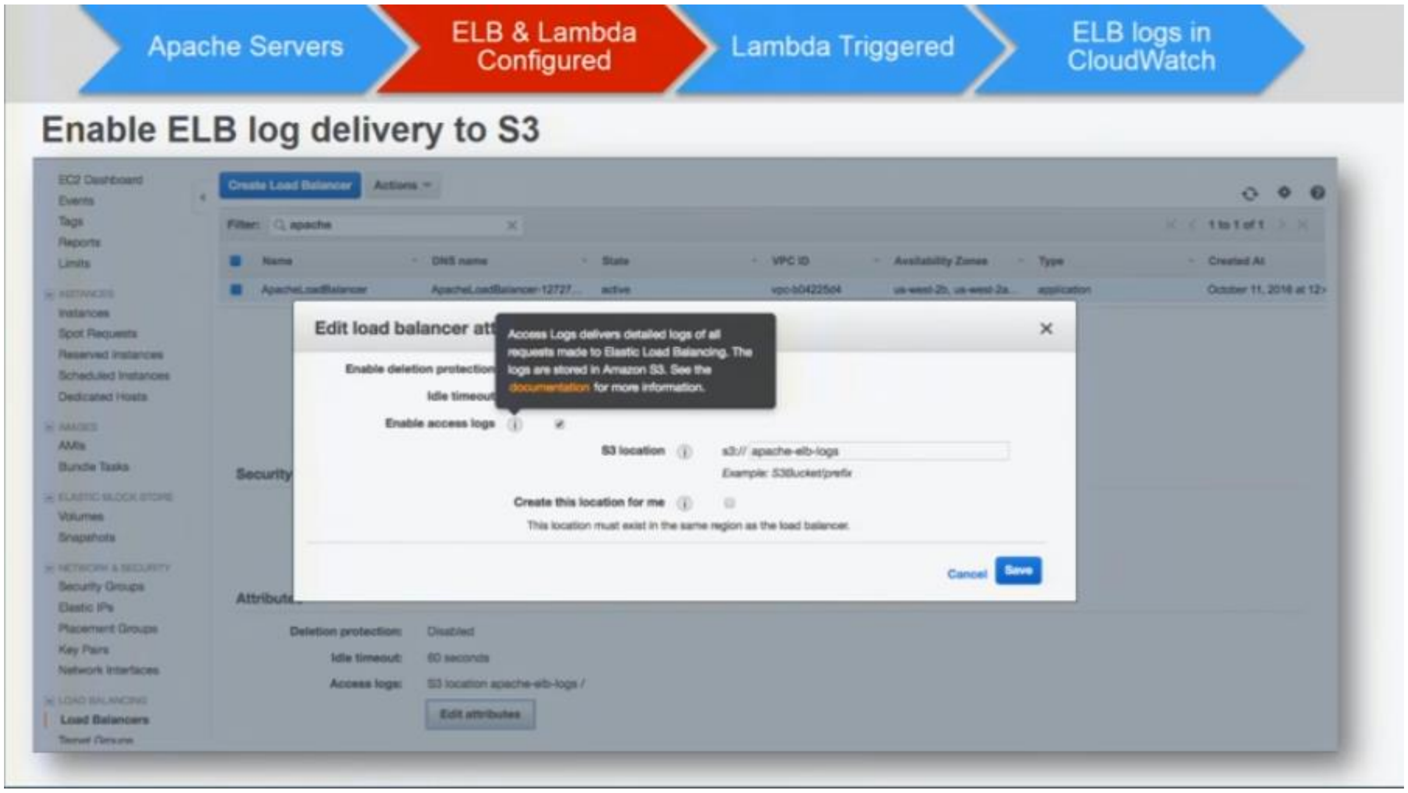
Registered instances

Instance ID	Name	Port	Availability Zone	Status
i-00255fd07ca832bd2	apache-2b	80	us-west-2b	healthy
i-0fa0adccbfc720442	apache-2c	80	us-west-2c	healthy
i-0250179af2a646134	apache-2a	80	us-west-2a	healthy

Availability Zones

Availability Zone	Instance Count	Healthy?

We then add those 3 apache servers to our ELB



Enable ELB log delivery to S3

Name	DNS name	State	VPC ID	Availability Zones	Type	Created At
ApacheLoadBalancer	ApacheLoadBalancer-12727...	active	vpc-b04225d4	us-west-2b, us-west-2a...	application	October 11, 2016 at 12:

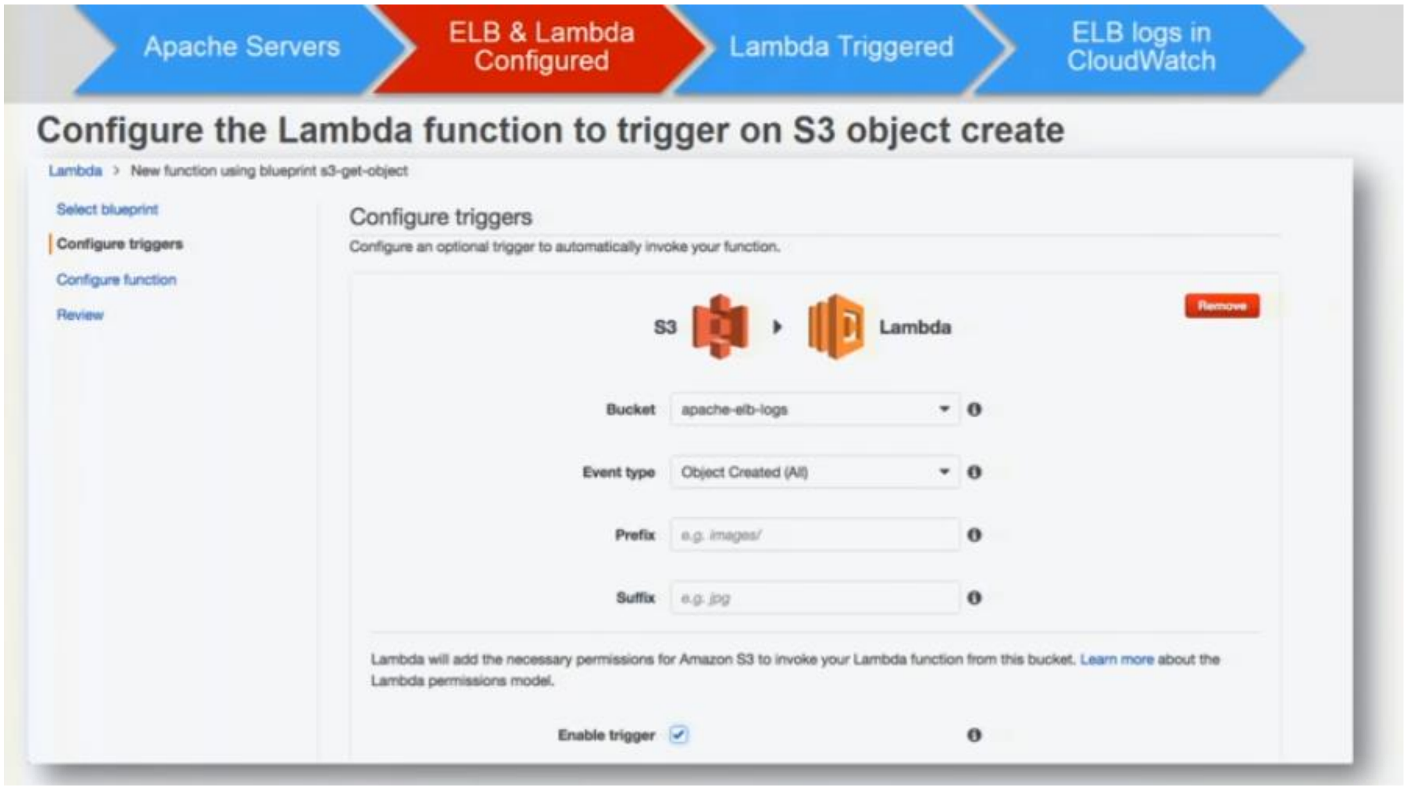
Edit load balancer att

Access Logs delivers detailed logs of all requests made to Elastic Load Balancing. The logs are stored in Amazon S3. See the documentation for more information.

Enable deletion protection
Idle timeout
Enable access logs ✓
S3 location s3:// apache-elb-logs
Example: S3Bucket/prefix
Create this location for me ☐
This location must exist in the same region as the load balancer.

Deletion protection: Disabled
Idle timeout: 60 seconds
Access logs: S3 location apache-elb-logs /

We then configure S3 delivery of the access logs to our S3 bucket



Configure the Lambda function to trigger on S3 object create

Lambda > New function using blueprint s3-get-object

Configure triggers

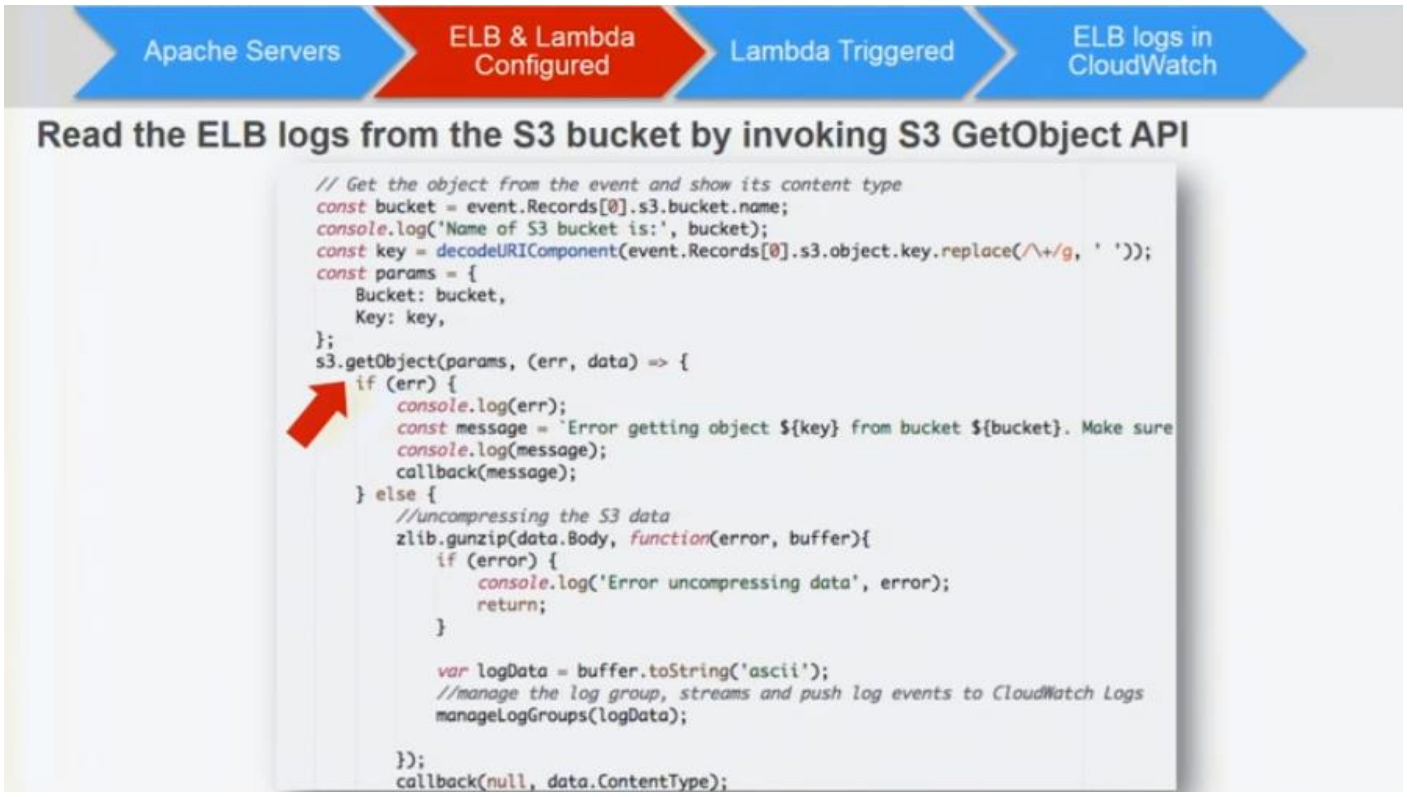
Configure an optional trigger to automatically invoke your function.

Bucket	apache-elb-logs
Event type	Object Created (All)
Prefix	e.g. images/
Suffix	e.g. jpg

Lambda will add the necessary permissions for Amazon S3 to invoke your Lambda function from this bucket. Learn more about the Lambda permissions model.

Enable trigger ✓

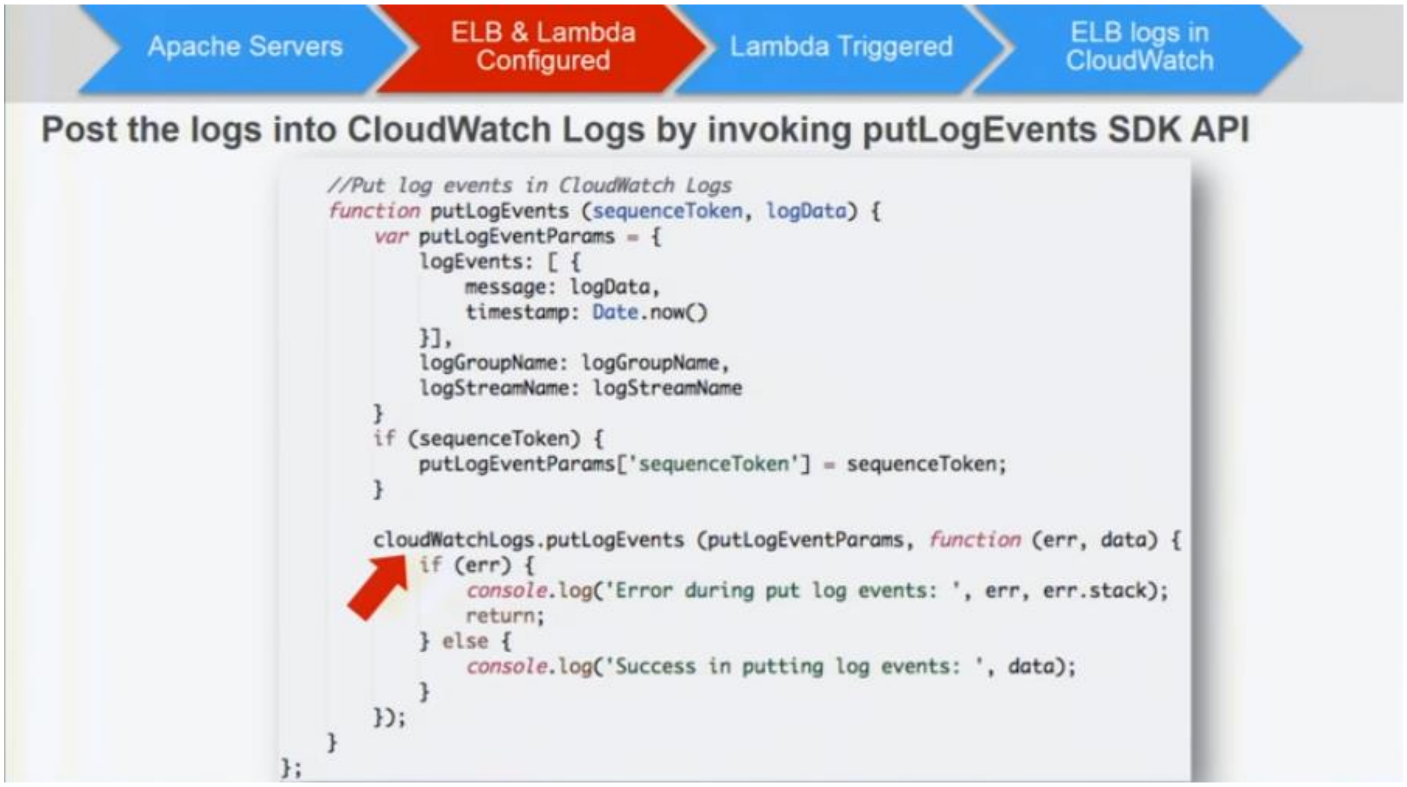
As those objects are being created, we have the option to set up a lambda function to trigger on the S3 object creation event



Read the ELB logs from the S3 bucket by invoking S3 GetObject API

```
// Get the object from the event and show its content type
const bucket = event.Records[0].s3.bucket.name;
console.log('Name of S3 bucket is:', bucket);
const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, ' '));
const params = {
    Bucket: bucket,
    Key: key,
};
s3.getObject(params, (err, data) => {
    if (err) {
        console.log(err);
        const message = `Error getting object ${key} from bucket ${bucket}. Make sure
        console.log(message);
        callback(message);
    } else {
        //uncompressing the S3 data
        zlib.gunzip(data.Body, function(error, buffer){
            if (error) {
                console.log('Error uncompressing data', error);
                return;
            }

            var logData = buffer.toString('ascii');
            //manage the log group, streams and push log events to CloudWatch Logs
            manageLogGroups(logData);

        });
        callback(null, data.ContentType);
```

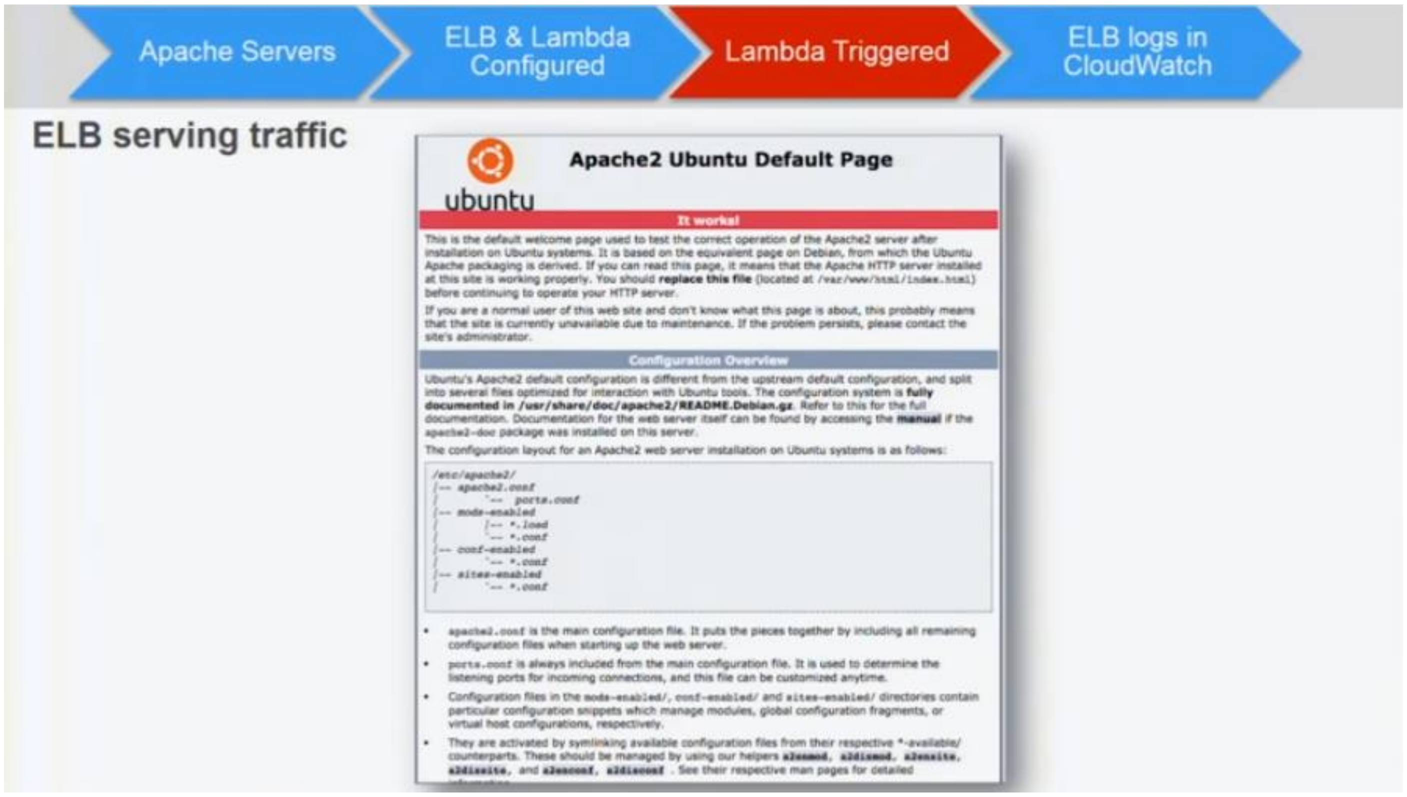
Our lambda function is going to get invoked when an object gets created in the S3 bucket, it then gets the object, decompresses it, break it up into pieces, then calls the AWS SDK's putLogEvents API for CloudWatch ***CW*** logs as below



Post the logs into CloudWatch Logs by invoking putLogEvents SDK API

```
    //Put log events in CloudWatch Logs
    function putLogEvents (sequenceToken, logData) {
        var putLogEventParams = {
            logEvents: [ {
                message: logData,
                timestamp: Date.now()
            }],
            logGroupName: logGroupName,
            logStreamName: logStreamName
        }
        if (sequenceToken) {
            putLogEventParams['sequenceToken'] = sequenceToken;
        }

        cloudWatchLogs.putLogEvents (putLogEventParams, function (err, data) {
            if (err) {
                console.log('Error during put log events: ', err, err.stack);
                return;
            } else {
                console.log('Success in putting log events: ', data);
            }
        });
    }
};
```

This ***putLogEvents*** API allows you to log from your apps directly into CW without first logging to a file.



ELB serving traffic

Apache2 Ubuntu Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at /var/www/html/index.html) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

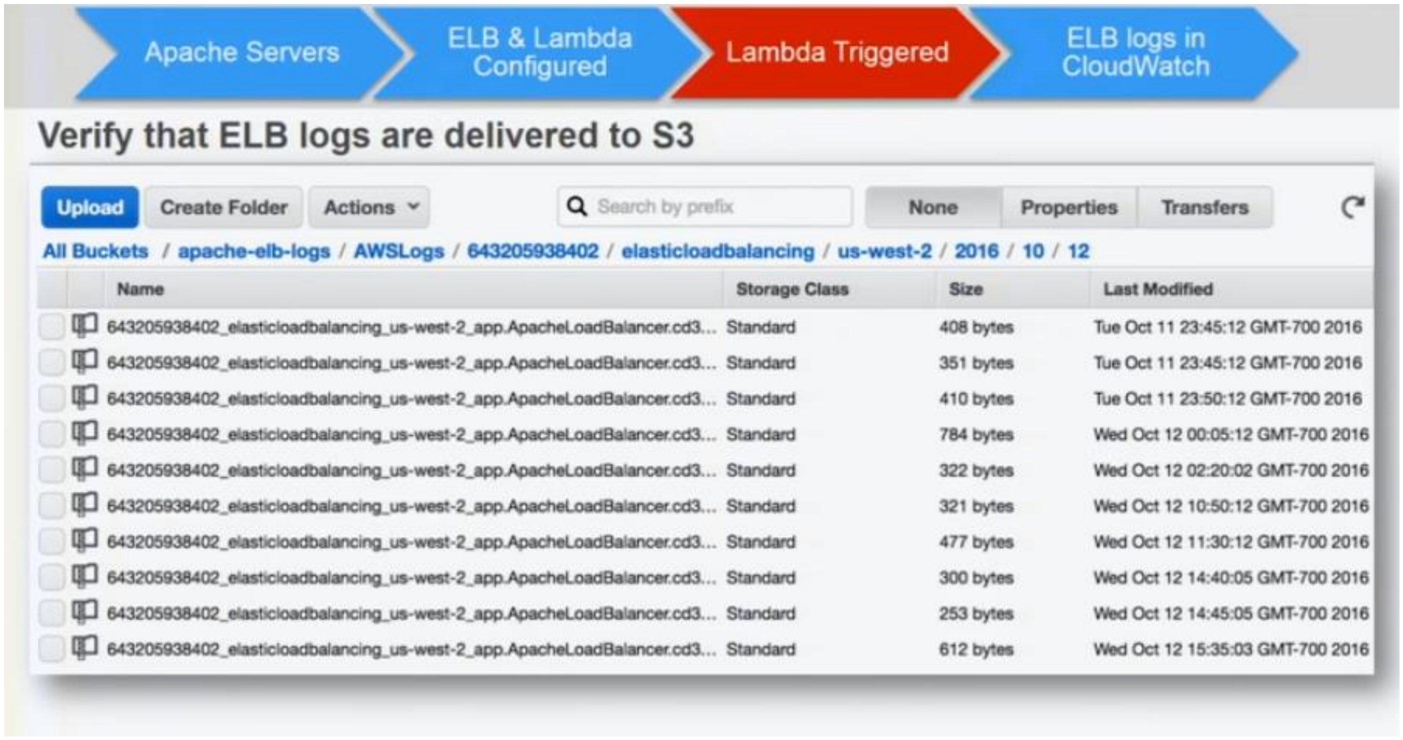
Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the apache2-doc package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

- apache2.conf is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
- ports.conf is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized anytime.
- Configuration files in the mods-enabled/, conf-enabled/ and sites-enabled/ directories contain particular configuration snippets which manage modules, global configuration fragments, or virtual host configurations, respectively.
- They are activated by symlinking available configuration files from their respective *-available/ counterparts. These should be managed by using our helpers **a2enmod**, **a2dismod**, **a2ensite**, **a2dissite**, and **a2enconf**, **a2disconf** . See their respective man pages for detailed

We can now test it out by visiting the ELK endpoint

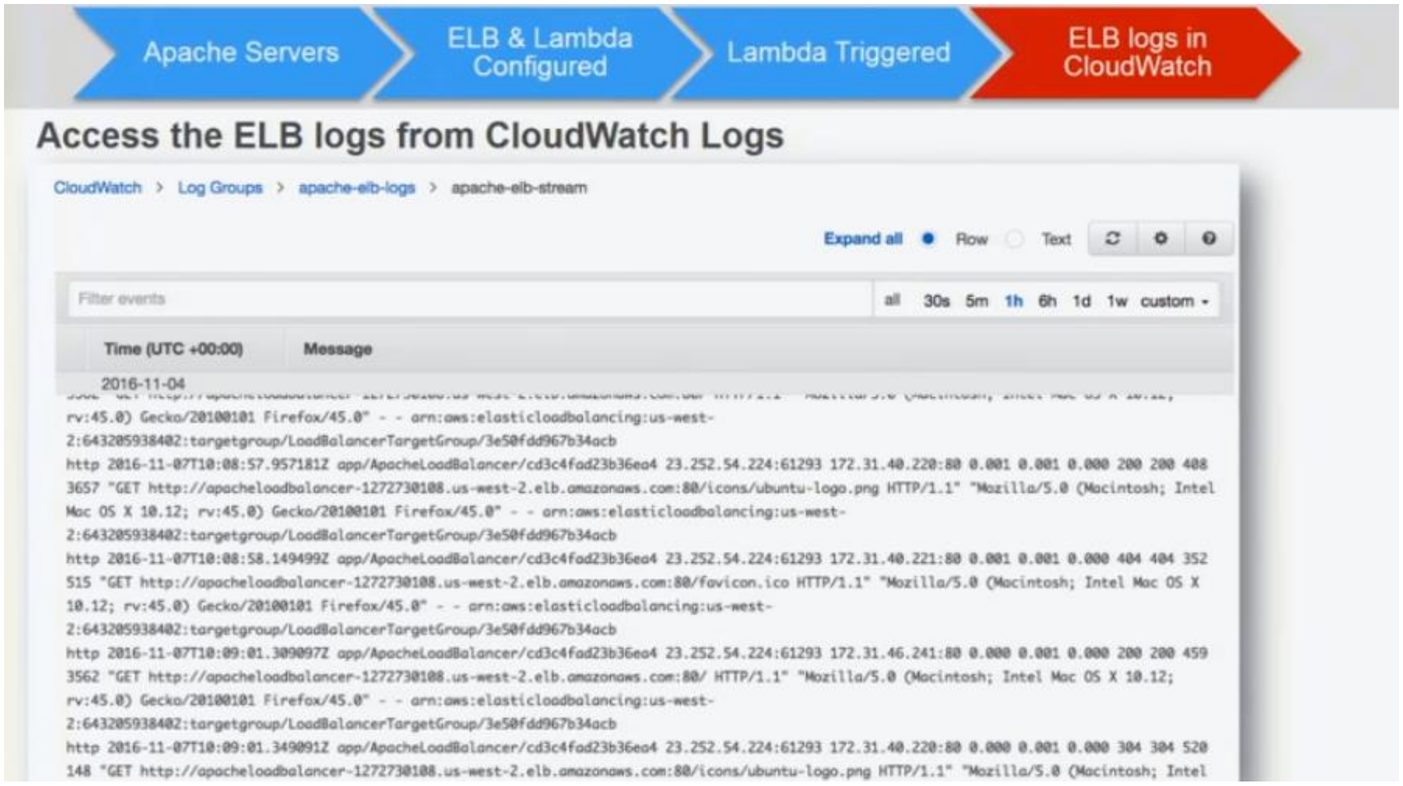


Verify that ELB logs are delivered to S3

All Buckets / apache-elb-logs / AWSLogs / 643205938402 / elasticloadbalancing / us-west-2 / 2016 / 10 / 12

Name	Storage Class	Size	Last Modified
643205938402_elasticloadbalancing_us-west-2_app.ApacheLoadBalancer.cd3...	Standard	408 bytes	Tue Oct 11 23:45:12 GMT-700 2016
643205938402_elasticloadbalancing_us-west-2_app.ApacheLoadBalancer.cd3...	Standard	351 bytes	Tue Oct 11 23:45:12 GMT-700 2016
643205938402_elasticloadbalancing_us-west-2_app.ApacheLoadBalancer.cd3...	Standard	410 bytes	Tue Oct 11 23:50:12 GMT-700 2016
643205938402_elasticloadbalancing_us-west-2_app.ApacheLoadBalancer.cd3...	Standard	784 bytes	Wed Oct 12 00:05:12 GMT-700 2016
643205938402_elasticloadbalancing_us-west-2_app.ApacheLoadBalancer.cd3...	Standard	322 bytes	Wed Oct 12 02:20:02 GMT-700 2016
643205938402_elasticloadbalancing_us-west-2_app.ApacheLoadBalancer.cd3...	Standard	321 bytes	Wed Oct 12 10:50:12 GMT-700 2016
643205938402_elasticloadbalancing_us-west-2_app.ApacheLoadBalancer.cd3...	Standard	477 bytes	Wed Oct 12 11:30:12 GMT-700 2016
643205938402_elasticloadbalancing_us-west-2_app.ApacheLoadBalancer.cd3...	Standard	300 bytes	Wed Oct 12 14:40:05 GMT-700 2016
643205938402_elasticloadbalancing_us-west-2_app.ApacheLoadBalancer.cd3...	Standard	253 bytes	Wed Oct 12 14:45:05 GMT-700 2016
643205938402_elasticloadbalancing_us-west-2_app.ApacheLoadBalancer.cd3...	Standard	612 bytes	Wed Oct 12 15:35:03 GMT-700 2016

We can start seeing the ELB logs getting delivered to our S3 bucket



Access the ELB logs from CloudWatch Logs

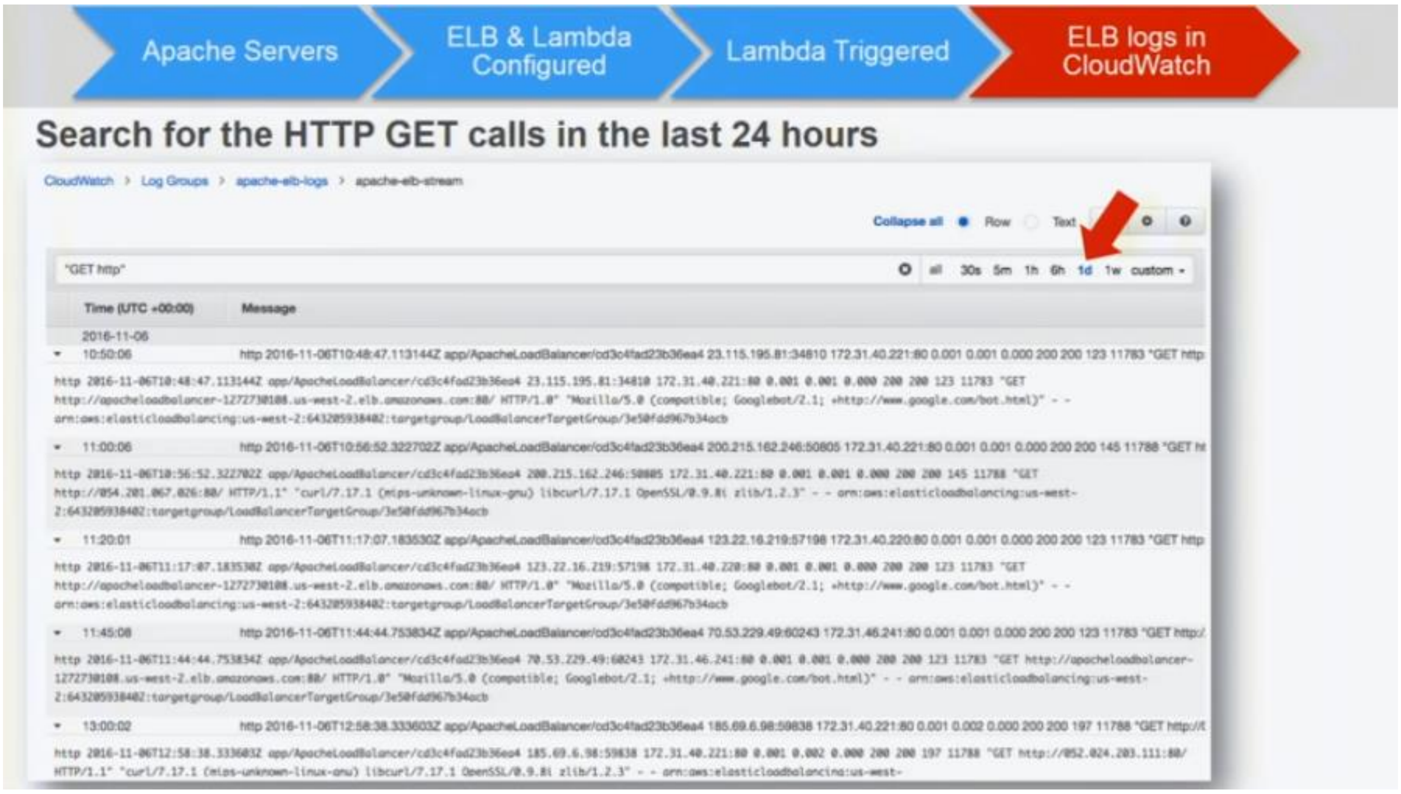
CloudWatch > Log Groups > apache-elb-logs > apache-elb-stream

Time (UTC +00:00) Message

2016-11-04

```
rv:45.0) Gecko/20100101 Firefox/45.0" - - arn:aws:elasticloadbalancing:us-west-
2:643205938402:targetgroup/LoadBalancerTargetGroup/3e50fdd967b34acb
http 2016-11-07T10:08:57.957181Z app/ApacheLoadBalancer/cd3c4fad23b36ea4 23.252.54.224:61293 172.31.40.220:80 0.001 0.001 0.000 200 200 408
3657 "GET http://apacheloadbalancer-1272730108.us-west-2.elb.amazonaws.com:80/icons/ubuntu-logo.png HTTP/1.1" "Mozilla/5.0 (Macintosh; Intel
Mac OS X 10.12; rv:45.0) Gecko/20100101 Firefox/45.0" - - arn:aws:elasticloadbalancing:us-west-
2:643205938402:targetgroup/LoadBalancerTargetGroup/3e50fdd967b34acb
http 2016-11-07T10:08:58.149499Z app/ApacheLoadBalancer/cd3c4fad23b36ea4 23.252.54.224:61293 172.31.40.221:80 0.001 0.001 0.000 404 404 352
515 "GET http://apacheloadbalancer-1272730108.us-west-2.elb.amazonaws.com:80/favicon.ico HTTP/1.1" "Mozilla/5.0 (Macintosh; Intel Mac OS X
10.12; rv:45.0) Gecko/20100101 Firefox/45.0" - - arn:aws:elasticloadbalancing:us-west-
2:643205938402:targetgroup/LoadBalancerTargetGroup/3e50fdd967b34acb
http 2016-11-07T10:09:01.309097Z app/ApacheLoadBalancer/cd3c4fad23b36ea4 23.252.54.224:61293 172.31.46.241:80 0.000 0.001 0.000 200 200 459
3562 "GET http://apacheloadbalancer-1272730108.us-west-2.elb.amazonaws.com:80/ HTTP/1.1" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12;
rv:45.0) Gecko/20100101 Firefox/45.0" - - arn:aws:elasticloadbalancing:us-west-
2:643205938402:targetgroup/LoadBalancerTargetGroup/3e50fdd967b34acb
http 2016-11-07T10:09:01.349091Z app/ApacheLoadBalancer/cd3c4fad23b36ea4 23.252.54.224:61293 172.31.40.220:80 0.000 0.001 0.000 304 304 520
148 "GET http://apacheloadbalancer-1272730108.us-west-2.elb.amazonaws.com:80/icons/ubuntu-logo.png HTTP/1.1" "Mozilla/5.0 (Macintosh; Intel
```

But because there is a lambda function being invoked in the background as the logs are getting created, we now get the data being streamed in near real time into our CW logs in a series

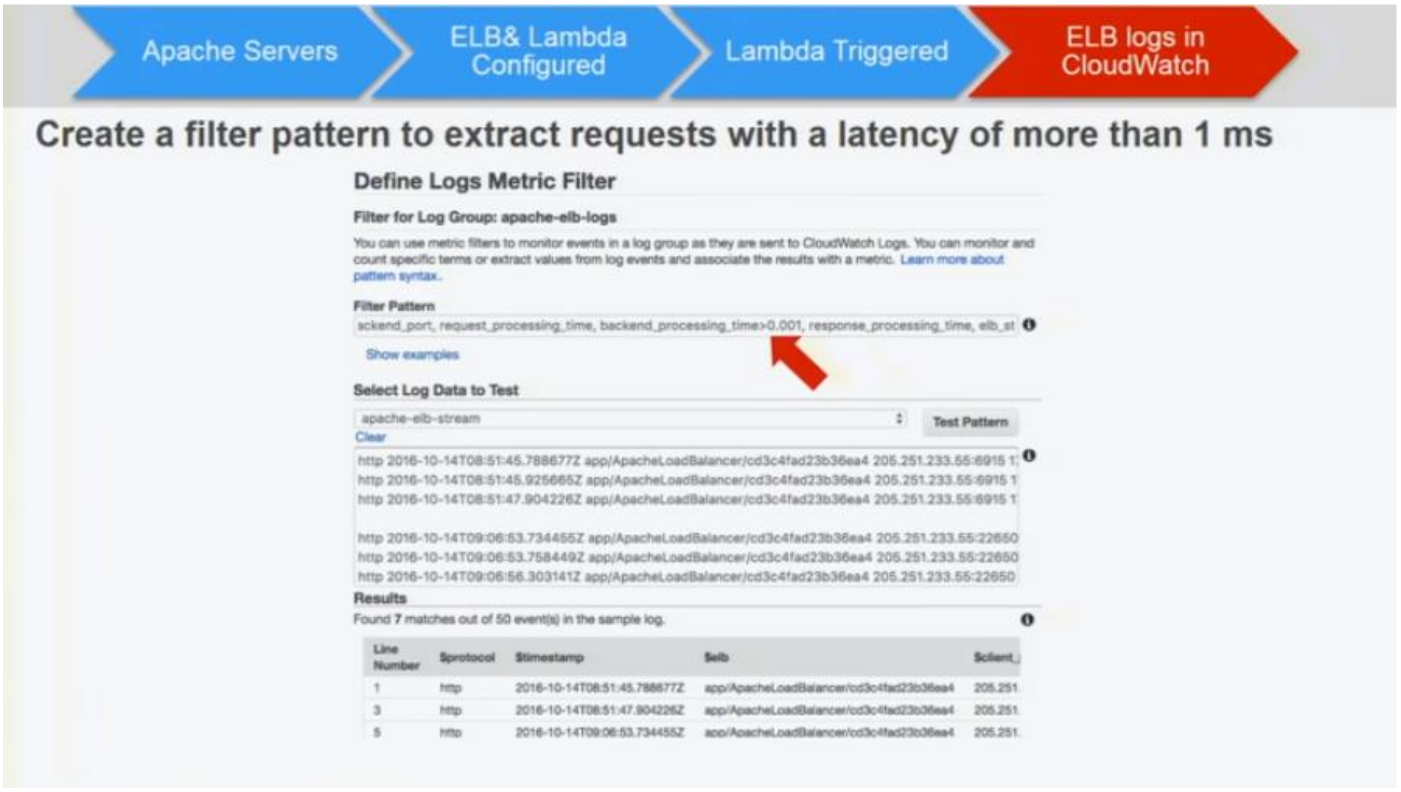

Apache Servers
ELB & Lambda Configured
Lambda Triggered
ELB logs in CloudWatch

Search for the HTTP GET calls in the last 24 hours

CloudWatch > Log Groups > apache-elb-logs > apache-elb-stream
Collapse all ● Row ○ Text
all 30s 5m 1h 6h 1d 1w custom

"GET http"

```
Time (UTC +00:00)   Message
2016-11-06
▾ 10:50:06   http 2016-11-06T10:48:47.113144Z app/ApacheLoadBalancer/cd3c4fad23b36ea4 23.115.195.81:34810 172.31.40.221:80 0.001 0.001 0.000 200 200 123 11783 "GET http
http 2016-11-06T10:48:47.113144Z app/ApacheLoadBalancer/cd3c4fad23b36ea4 23.115.195.81:34810 172.31.40.221:80 0.001 0.001 0.000 200 200 123 11783 "GET
http://apacheloadbalancer-1272730108.us-west-2.elb.amazonaws.com:80/ HTTP/1.0" "Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)" - -
arn:aws:elasticloadbalancing:us-west-2:643205938402:targetgroup/LoadBalancerTargetGroup/3e50fdd967b34acb
▾ 11:00:06   http 2016-11-06T10:56:52.322702Z app/ApacheLoadBalancer/cd3c4fad23b36ea4 200.215.162.246:50805 172.31.40.221:80 0.001 0.001 0.000 200 200 145 11788 "GET ht
http 2016-11-06T10:56:52.322702Z app/ApacheLoadBalancer/cd3c4fad23b36ea4 200.215.162.246:50805 172.31.40.221:80 0.001 0.001 0.000 200 200 145 11788 "GET
http://054.201.067.026:80/ HTTP/1.1" "curl/7.17.1 (mips-unknown-linux-gnu) libcurl/7.17.1 OpenSSL/0.9.8i zlib/1.2.3" - - arn:aws:elasticloadbalancing:us-west-
2:643205938402:targetgroup/LoadBalancerTargetGroup/3e50fdd967b34acb
▾ 11:20:01   http 2016-11-06T11:17:07.183530Z app/ApacheLoadBalancer/cd3c4fad23b36ea4 123.22.16.219:57198 172.31.40.220:80 0.001 0.001 0.000 200 200 123 11783 "GET http
http 2016-11-06T11:17:07.183530Z app/ApacheLoadBalancer/cd3c4fad23b36ea4 123.22.16.219:57198 172.31.40.220:80 0.001 0.001 0.000 200 200 123 11783 "GET
http://apacheloadbalancer-1272730108.us-west-2.elb.amazonaws.com:80/ HTTP/1.0" "Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)" - -
arn:aws:elasticloadbalancing:us-west-2:643205938402:targetgroup/LoadBalancerTargetGroup/3e50fdd967b34acb
▾ 11:45:08   http 2016-11-06T11:44:44.753834Z app/ApacheLoadBalancer/cd3c4fad23b36ea4 70.53.229.49:60243 172.31.46.241:80 0.001 0.001 0.000 200 200 123 11783 "GET http:/
http 2016-11-06T11:44:44.753834Z app/ApacheLoadBalancer/cd3c4fad23b36ea4 70.53.229.49:60243 172.31.46.241:80 0.001 0.001 0.000 200 200 123 11783 "GET http://apacheloadbalancer-
1272730108.us-west-2.elb.amazonaws.com:80/ HTTP/1.0" "Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)" - - arn:aws:elasticloadbalancing:us-west-
2:643205938402:targetgroup/LoadBalancerTargetGroup/3e50fdd967b34acb
▾ 13:00:02   http 2016-11-06T12:58:38.333603Z app/ApacheLoadBalancer/cd3c4fad23b36ea4 185.69.6.98:59838 172.31.40.221:80 0.001 0.002 0.000 200 200 197 11788 "GET http://(
http 2016-11-06T12:58:38.333603Z app/ApacheLoadBalancer/cd3c4fad23b36ea4 185.69.6.98:59838 172.31.40.221:80 0.001 0.002 0.000 200 200 197 11788 "GET http://052.024.203.111:80/
HTTP/1.1" "curl/7.17.1 (mips-unknown-linux-gnu) libcurl/7.17.1 OpenSSL/0.9.8i zlib/1.2.3" - - arn:aws:elasticloadbalancing:us-west-
```



Create a filter pattern to extract requests with a latency of more than 1 ms

Define Logs Metric Filter

Filter for Log Group: apache-elb-logs

You can use metric filters to monitor events in a log group as they are sent to CloudWatch Logs. You can monitor and count specific terms or extract values from log events and associate the results with a metric. Learn more about pattern syntax.

Filter Pattern

```
ackend_port, request_processing_time, backend_processing_time>0.001, response_processing_time, elb_st
```

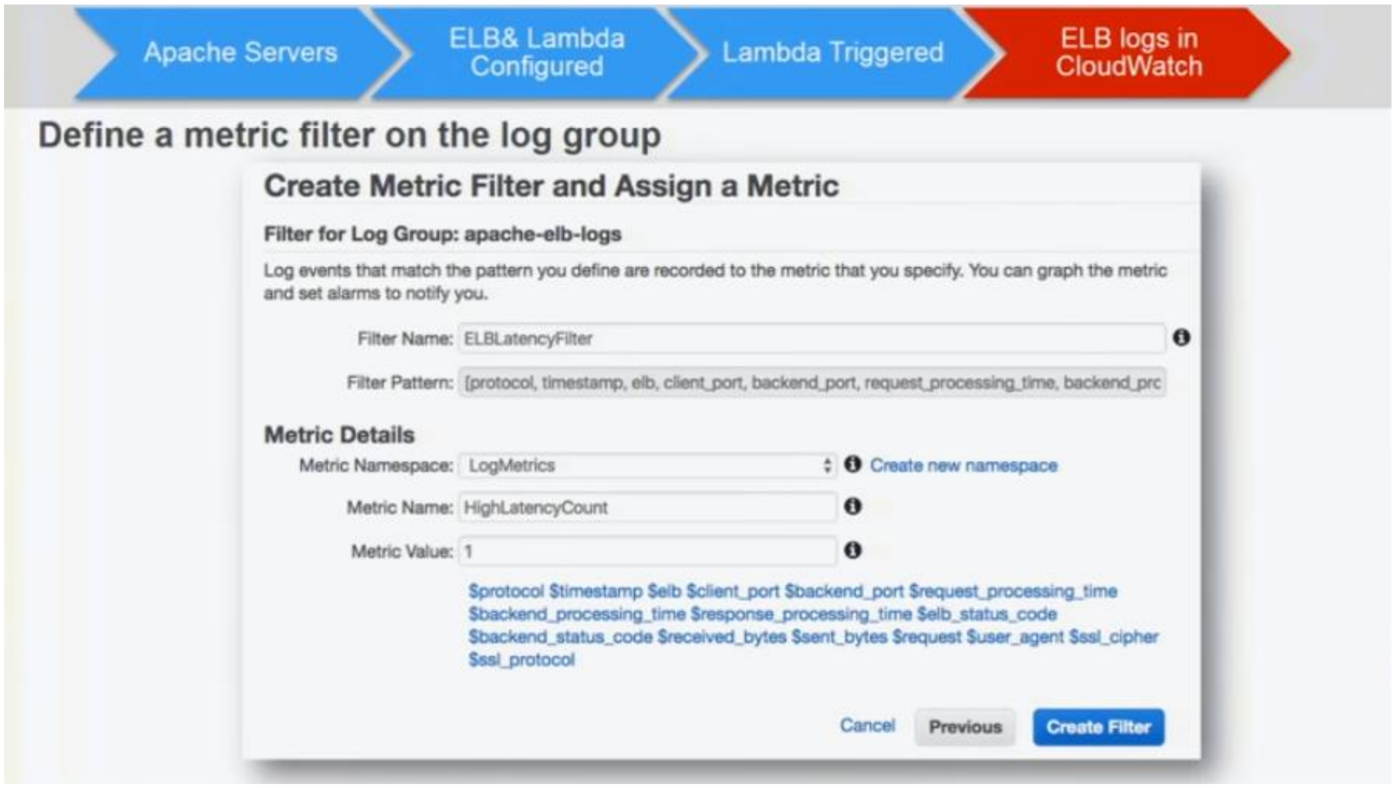
Select Log Data to Test

```
http 2016-10-14T08:51:45.788677Z app/ApacheLoadBalancer/cd3c4fad23b36ea4 205.251.233.55:6915 1:
http 2016-10-14T08:51:45.925665Z app/ApacheLoadBalancer/cd3c4fad23b36ea4 205.251.233.55:6915 1
http 2016-10-14T08:51:47.904226Z app/ApacheLoadBalancer/cd3c4fad23b36ea4 205.251.233.55:6915 1

http 2016-10-14T09:06:53.734455Z app/ApacheLoadBalancer/cd3c4fad23b36ea4 205.251.233.55:22650
http 2016-10-14T09:06:53.758449Z app/ApacheLoadBalancer/cd3c4fad23b36ea4 205.251.233.55:22650
http 2016-10-14T09:06:56.303141Z app/ApacheLoadBalancer/cd3c4fad23b36ea4 205.251.233.55:22650
```

Results

Found **7** matches out of 50 event(s) in the sample log.

Line Number	$protocol	$timestamp	$elb	$client_
1	http	2016-10-14T08:51:45.788677Z	app/ApacheLoadBalancer/cd3c4fad23b36ea4	205.251.
3	http	2016-10-14T08:51:47.904226Z	app/ApacheLoadBalancer/cd3c4fad23b36ea4	205.251.
5	http	2016-10-14T09:06:53.734455Z	app/ApacheLoadBalancer/cd3c4fad23b36ea4	205.251.



Define a metric filter on the log group

Create Metric Filter and Assign a Metric

Filter for Log Group: apache-elb-logs

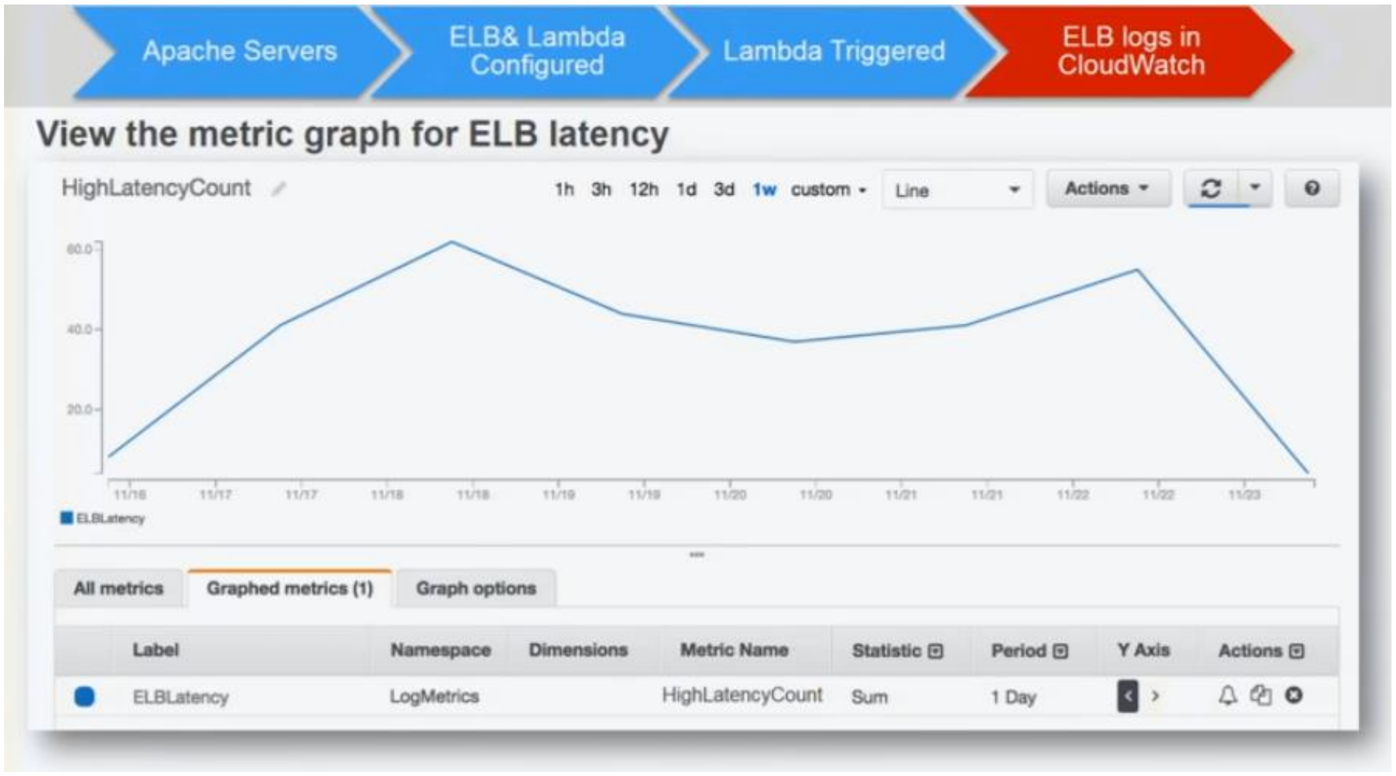
Log events that match the pattern you define are recorded to the metric that you specify. You can graph the metric and set alarms to notify you.

Filter Name:	ELBLatencyFilter
Filter Pattern:	[protocol, timestamp, elb, client_port, backend_port, request_processing_time, backend_prc

Metric Details

Metric Namespace:	LogMetrics
Metric Name:	HighLatencyCount
Metric Value:	1

$protocol $timestamp $elb $client_port $backend_port $request_processing_time $backend_processing_time $response_processing_time $elb_status_code $backend_status_code $received_bytes $sent_bytes $request $user_agent $ssl_cipher $ssl_protocol



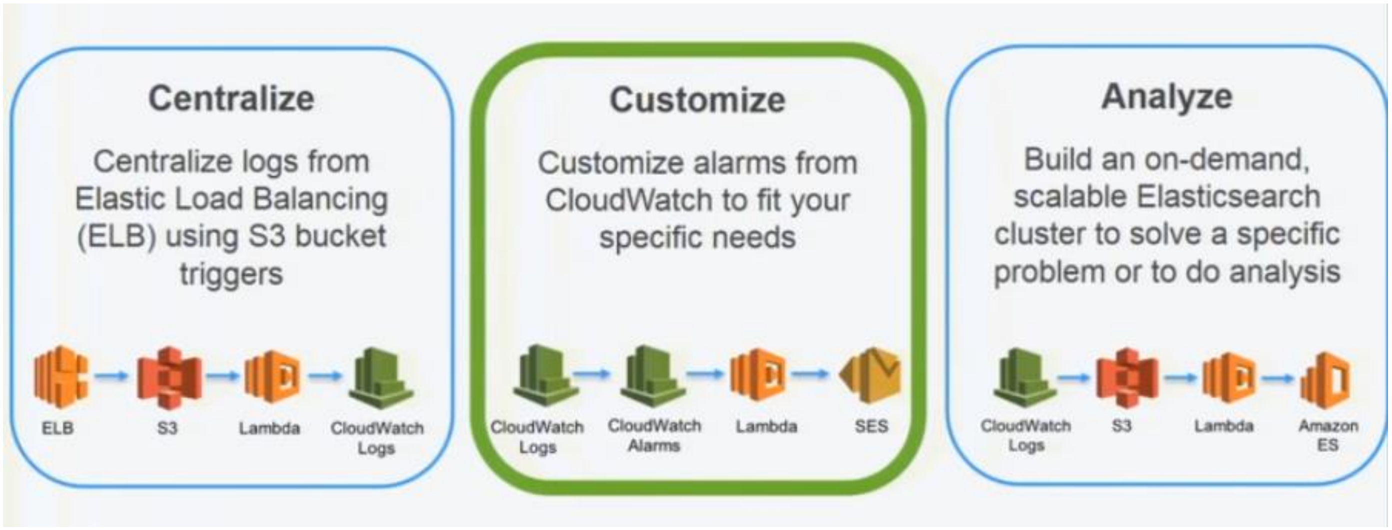
View the metric graph for ELB latency

	Label	Namespace	Dimensions	Metric Name	Statistic	Period	Y Axis	Actions
●	ELBLatency	LogMetrics		HighLatencyCount	Sum	1 Day		

Key takeaways

1. S3 delivered log data from any source can be centralized into CloudWatch Logs using Lambda

2. You can search and extract metrics from those logs in near real time



Centralize

Centralize logs from Elastic Load Balancing (ELB) using S3 bucket triggers

Customize

Customize alarms from CloudWatch to fit your specific needs

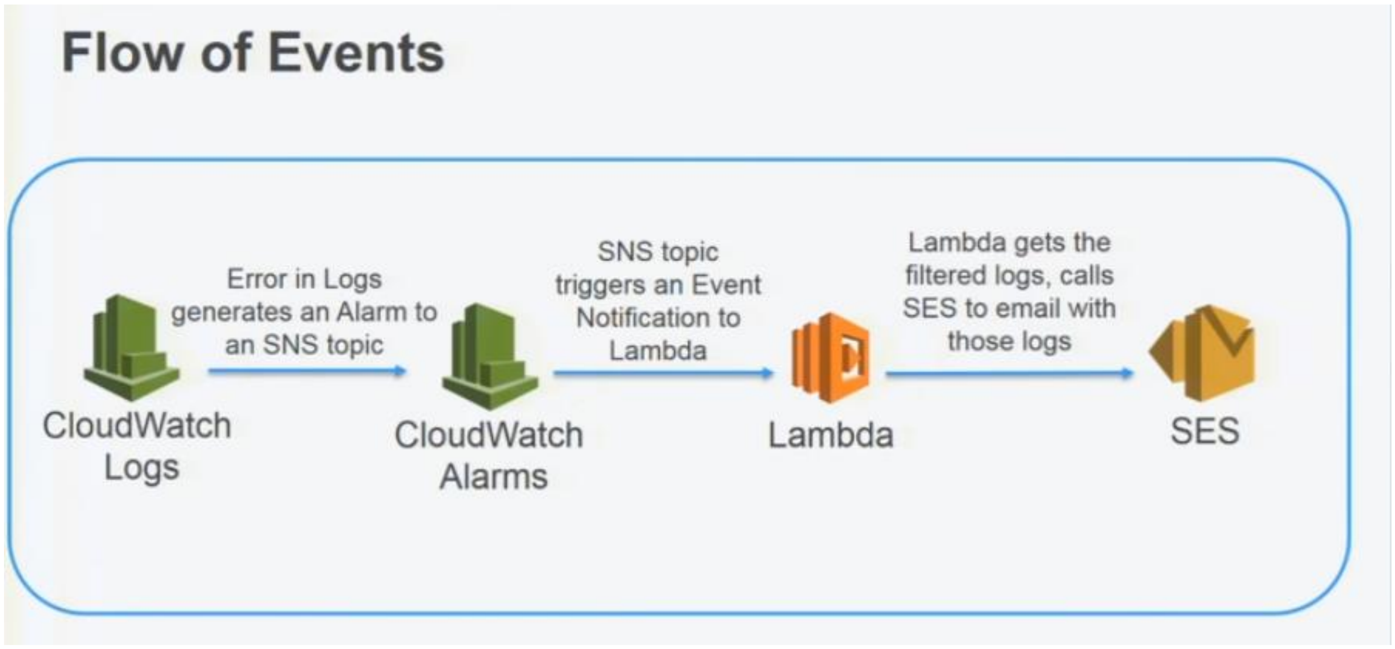
Analyze

Build an on-demand, scalable Elasticsearch cluster to solve a specific problem or to do analysis

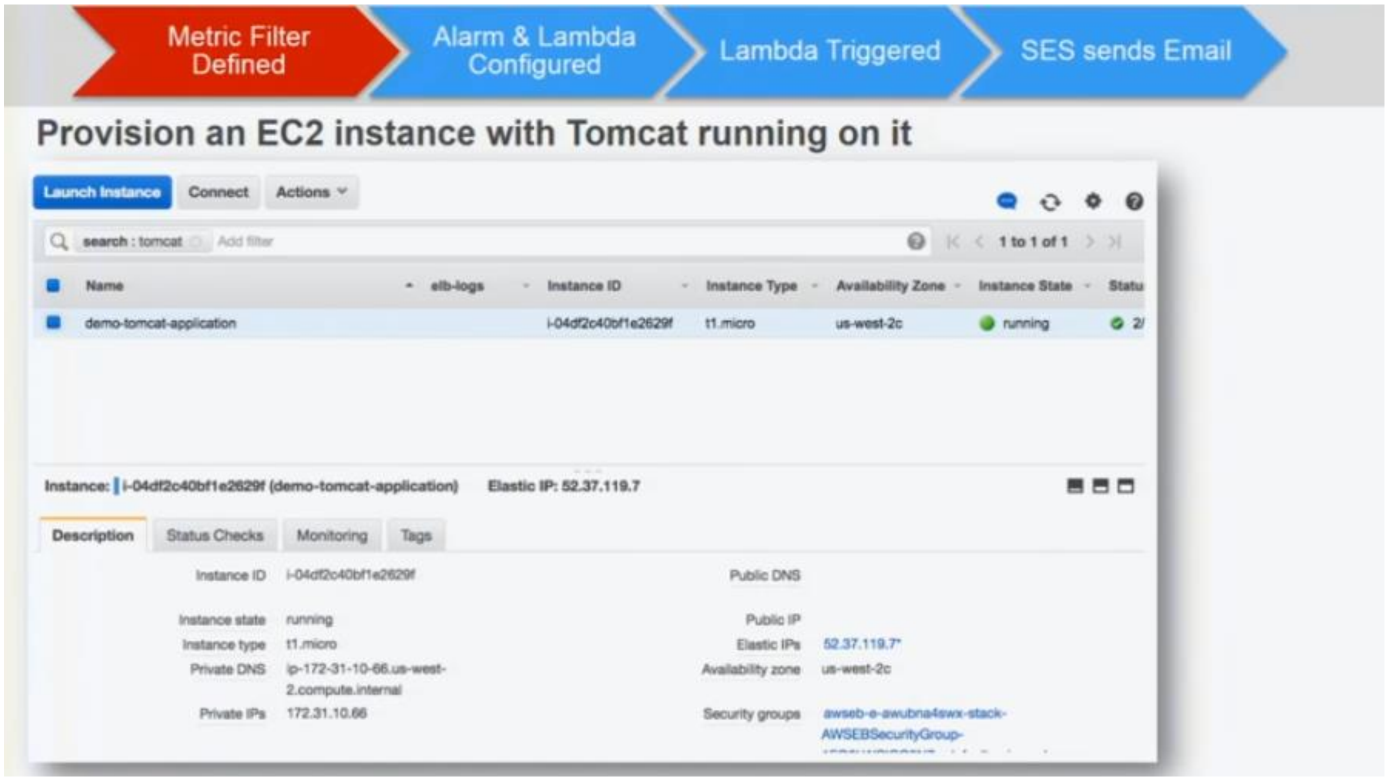
Problem Statements

- When you get an alarm you want enough information to decide whether it needs immediate attention or not

- You want to customize the alarm text and format to your operational needs


Flow of Events
Error in Logs generates an Alarm to an SNS topic
SNS topic triggers an Event Notification to Lambda
Lambda gets the filtered logs, calls SES to email with those logs
CloudWatch Logs
CloudWatch Alarms
Lambda
SES

We can send generated CW error logs to trigger an alarm and published to an SNS topic, the SNS topic will then trigger an event notification to a lambda, the lambda function then pulls the information out of the alarm, use the information to search the log service for the SLA breach emitted from the log service and invoke SES with the logs to forward to the operator email in rich text format what to do



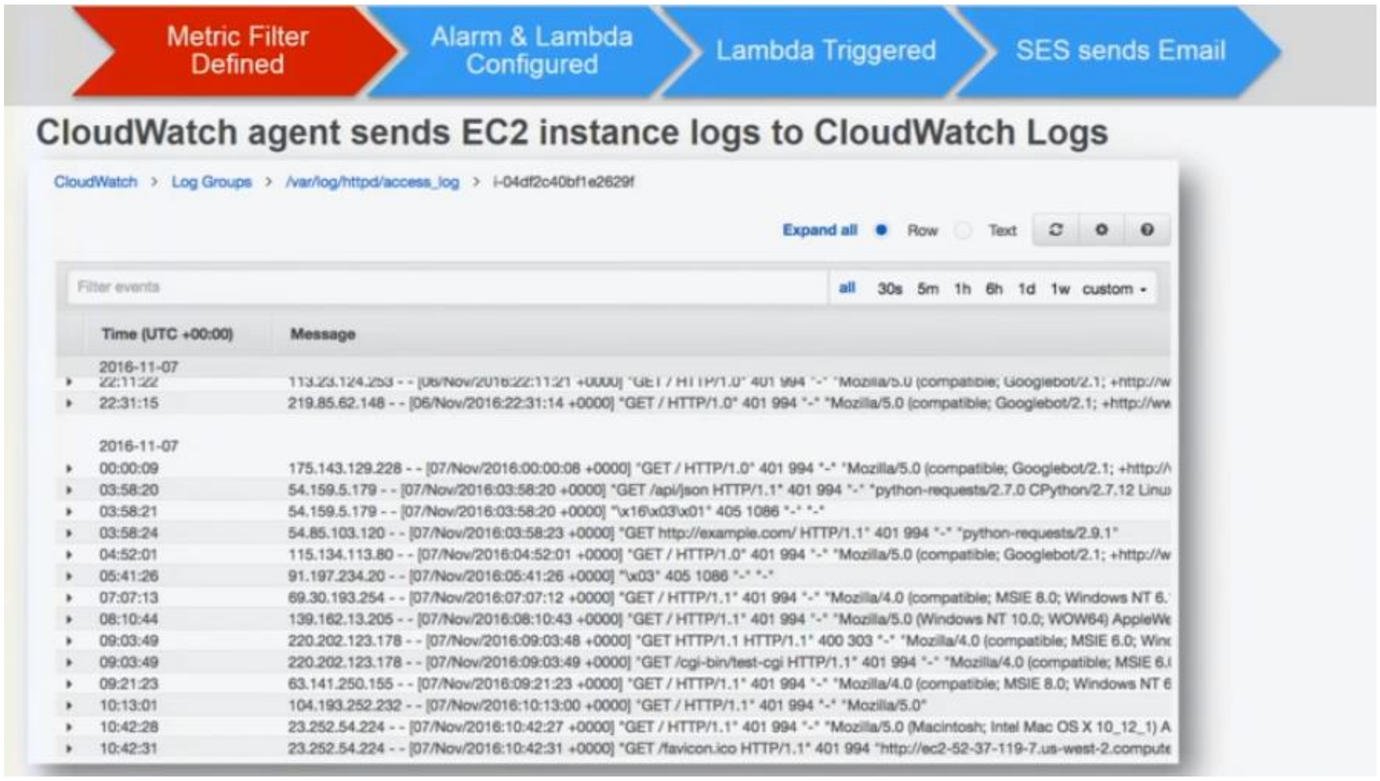
Provision an EC2 instance with Tomcat running on it

search : tomcat   Add filter   1 to 1 of 1

Name	elb-logs	Instance ID	Instance Type	Availability Zone	Instance State	Statu
demo-tomcat-application		i-04df2c40bf1e2629f	t1.micro	us-west-2c	● running	✓ 2/

Instance: i-04df2c40bf1e2629f (demo-tomcat-application)   Elastic IP: 52.37.119.7

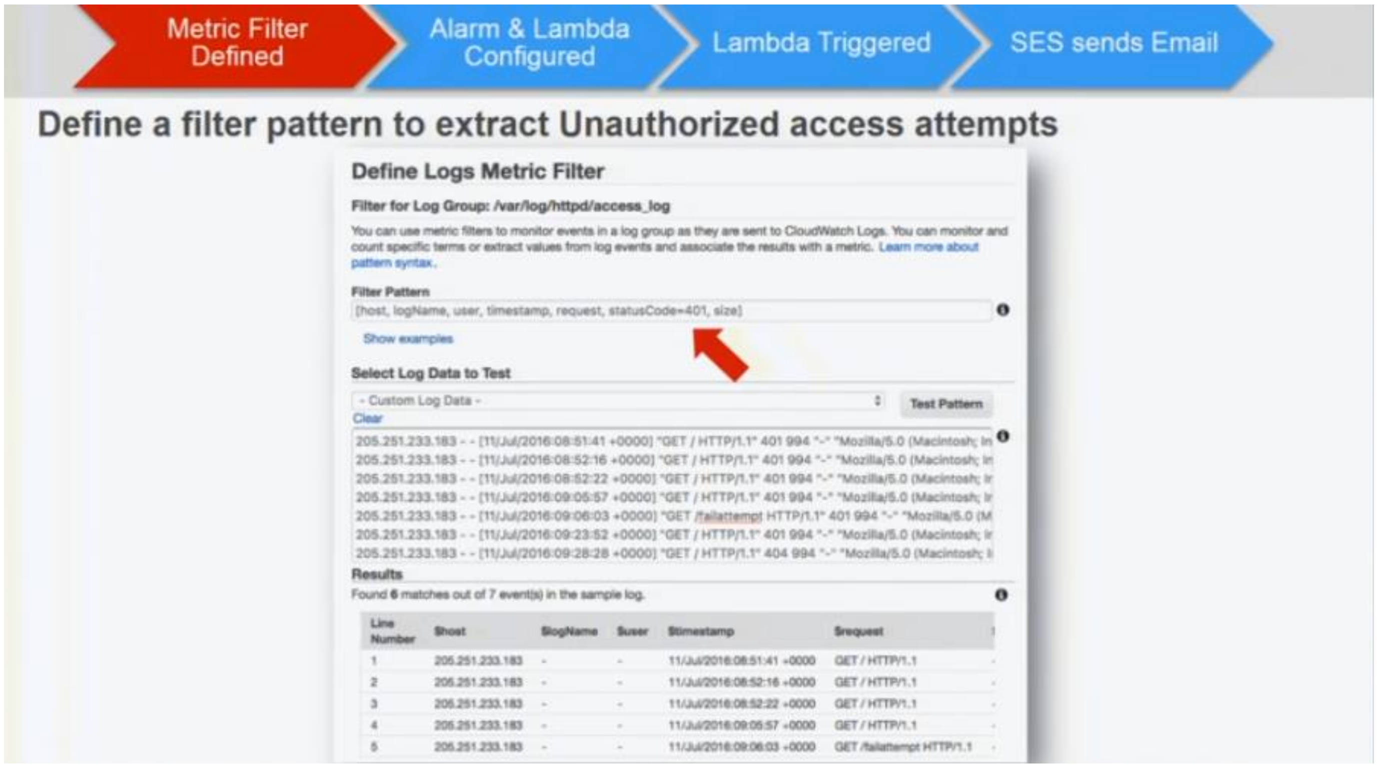
Instance ID	i-04df2c40bf1e2629f	Public DNS	
Instance state	running	Public IP	
Instance type	t1.micro	Elastic IPs	52.37.119.7*
Private DNS	ip-172-31-10-66.us-west-2.compute.internal	Availability zone	us-west-2c
Private IPs	172.31.10.66	Security groups	awseb-e-awubna4swx-stack-AWSEBSecurityGroup-



CloudWatch agent sends EC2 instance logs to CloudWatch Logs

CloudWatch > Log Groups > /var/log/httpd/access_log > i-04df2c40bf1e2629f

```
Time (UTC +00:00)   Message
2016-11-07
▸ 22:11:22          113.23.124.253 - - [06/Nov/2016:22:11:21 +0000] "GET / HTTP/1.0" 401 994 "-" "Mozilla/5.0 (compatible; Googlebot/2.1; +http://w
▸ 22:31:15          219.85.62.148 - - [06/Nov/2016:22:31:14 +0000] "GET / HTTP/1.0" 401 994 "-" "Mozilla/5.0 (compatible; Googlebot/2.1; +http://ww

2016-11-07
▸ 00:00:09          175.143.129.228 - - [07/Nov/2016:00:00:08 +0000] "GET / HTTP/1.0" 401 994 "-" "Mozilla/5.0 (compatible; Googlebot/2.1; +http://
▸ 03:58:20          54.159.5.179 - - [07/Nov/2016:03:58:20 +0000] "GET /api/json HTTP/1.1" 401 994 "-" "python-requests/2.7.0 CPython/2.7.12 Linu
▸ 03:58:21          54.159.5.179 - - [07/Nov/2016:03:58:20 +0000] "\x16\x03\x01" 405 1086 "-" "-"
▸ 03:58:24          54.85.103.120 - - [07/Nov/2016:03:58:23 +0000] "GET http://example.com/ HTTP/1.1" 401 994 "-" "python-requests/2.9.1"
▸ 04:52:01          115.134.113.80 - - [07/Nov/2016:04:52:01 +0000] "GET / HTTP/1.0" 401 994 "-" "Mozilla/5.0 (compatible; Googlebot/2.1; +http://w
▸ 05:41:26          91.197.234.20 - - [07/Nov/2016:05:41:26 +0000] "\x03" 405 1086 "-" "-"
▸ 07:07:13          69.30.193.254 - - [07/Nov/2016:07:07:12 +0000] "GET / HTTP/1.1" 401 994 "-" "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.
▸ 08:10:44          139.162.13.205 - - [07/Nov/2016:08:10:43 +0000] "GET / HTTP/1.1" 401 994 "-" "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWe
▸ 09:03:49          220.202.123.178 - - [07/Nov/2016:09:03:48 +0000] "GET HTTP/1.1 HTTP/1.1" 400 303 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Win
▸ 09:03:49          220.202.123.178 - - [07/Nov/2016:09:03:49 +0000] "GET /cgi-bin/test-cgi HTTP/1.1" 401 994 "-" "Mozilla/4.0 (compatible; MSIE 6.
▸ 09:21:23          63.141.250.155 - - [07/Nov/2016:09:21:23 +0000] "GET / HTTP/1.1" 401 994 "-" "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6
▸ 10:13:01          104.193.252.232 - - [07/Nov/2016:10:13:00 +0000] "GET / HTTP/1.1" 401 994 "-" "Mozilla/5.0"
▸ 10:42:28          23.252.54.224 - - [07/Nov/2016:10:42:27 +0000] "GET / HTTP/1.1" 401 994 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_1) A
▸ 10:42:31          23.252.54.224 - - [07/Nov/2016:10:42:31 +0000] "GET /favicon.ico HTTP/1.1" 401 994 "http://ec2-52-37-119-7.us-west-2.compute
```



Define a filter pattern to extract Unauthorized access attempts

Define Logs Metric Filter

Filter for Log Group: /var/log/httpd/access_log

You can use metric filters to monitor events in a log group as they are sent to CloudWatch Logs. You can monitor and count specific terms or extract values from log events and associate the results with a metric. Learn more about pattern syntax.

Filter Pattern

```
[host, logName, user, timestamp, request, statusCode=401, size]
```

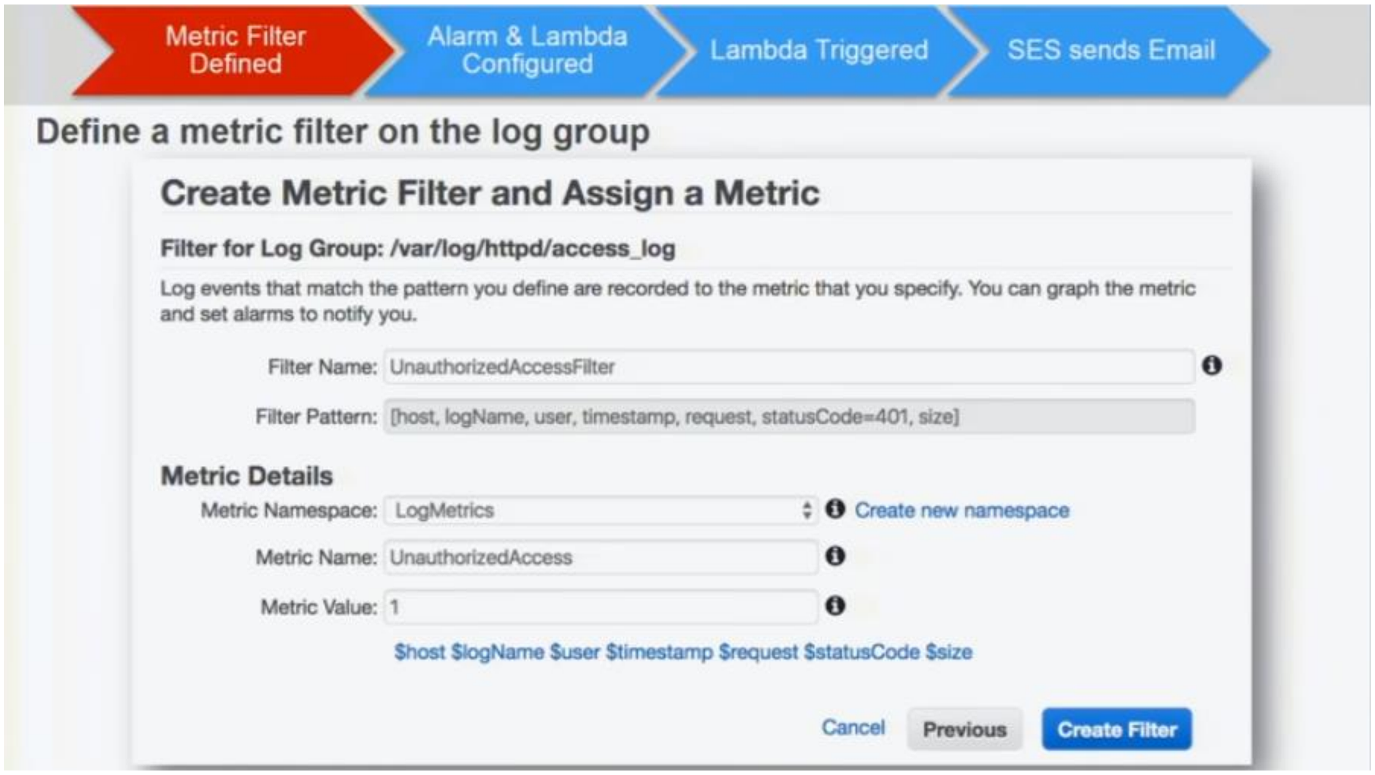
Select Log Data to Test

```
205.251.233.183 - - [11/Jul/2016:08:51:41 +0000] "GET / HTTP/1.1" 401 994 "-" "Mozilla/5.0 (Macintosh; In
205.251.233.183 - - [11/Jul/2016:08:52:16 +0000] "GET / HTTP/1.1" 401 994 "-" "Mozilla/5.0 (Macintosh; In
205.251.233.183 - - [11/Jul/2016:08:52:22 +0000] "GET / HTTP/1.1" 401 994 "-" "Mozilla/5.0 (Macintosh; Ir
205.251.233.183 - - [11/Jul/2016:09:05:57 +0000] "GET / HTTP/1.1" 401 994 "-" "Mozilla/5.0 (Macintosh; Ir
205.251.233.183 - - [11/Jul/2016:09:06:03 +0000] "GET /failattempt HTTP/1.1" 401 994 "-" "Mozilla/5.0 (M
205.251.233.183 - - [11/Jul/2016:09:23:52 +0000] "GET / HTTP/1.1" 401 994 "-" "Mozilla/5.0 (Macintosh; Ir
205.251.233.183 - - [11/Jul/2016:09:28:28 +0000] "GET / HTTP/1.1" 404 994 "-" "Mozilla/5.0 (Macintosh; I
```

Results

Found **6** matches out of 7 event(s) in the sample log.

Line Number	$host	$logName	$user	$timestamp	$request	
1	205.251.233.183	-	-	11/Jul/2016:08:51:41 +0000	GET / HTTP/1.1	-
2	205.251.233.183	-	-	11/Jul/2016:08:52:16 +0000	GET / HTTP/1.1	-
3	205.251.233.183	-	-	11/Jul/2016:08:52:22 +0000	GET / HTTP/1.1	-
4	205.251.233.183	-	-	11/Jul/2016:09:05:57 +0000	GET / HTTP/1.1	-
5	205.251.233.183	-	-	11/Jul/2016:09:06:03 +0000	GET /failattempt HTTP/1.1	-



Define a metric filter on the log group

Create Metric Filter and Assign a Metric

Filter for Log Group: /var/log/httpd/access_log

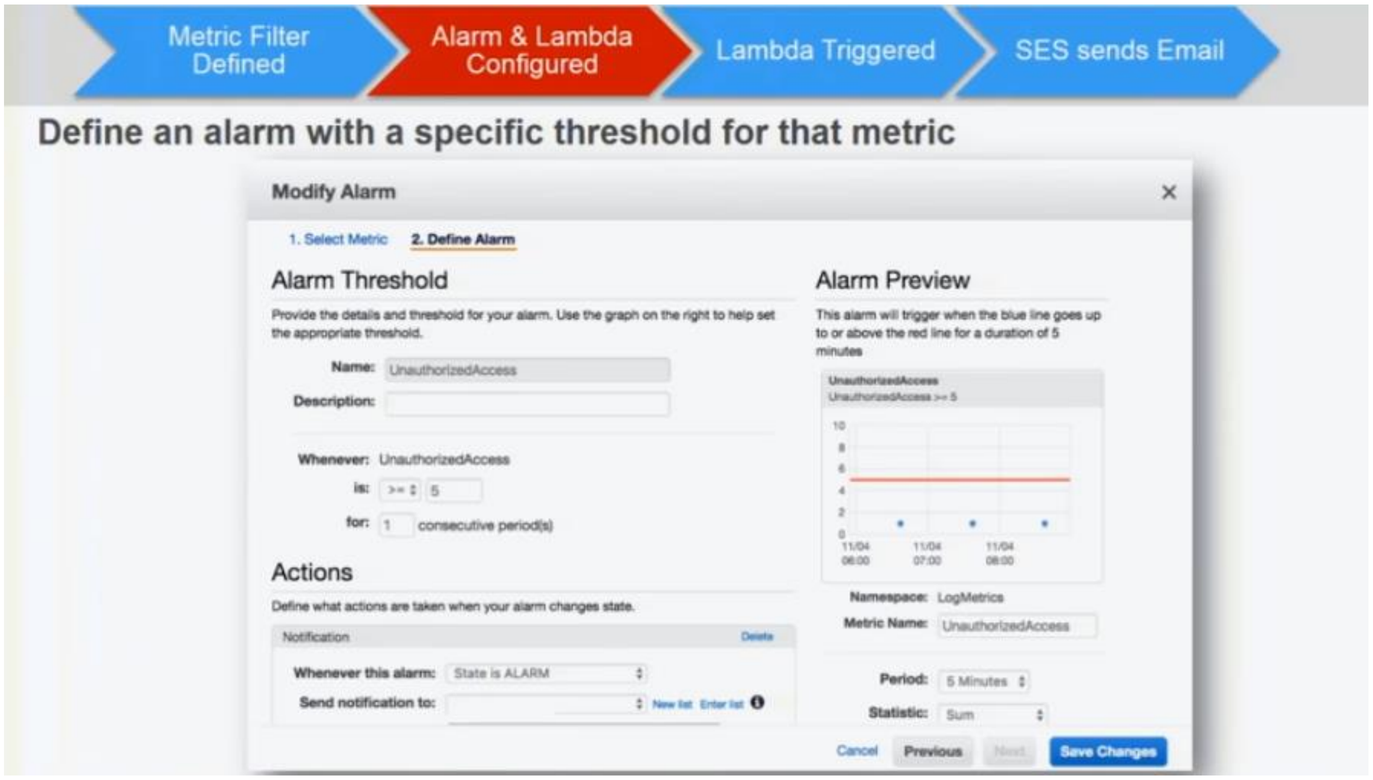
Log events that match the pattern you define are recorded to the metric that you specify. You can graph the metric and set alarms to notify you.

Filter Name:	UnauthorizedAccessFilter
Filter Pattern:	[host, logName, user, timestamp, request, statusCode=401, size]

Metric Details

Metric Namespace:	LogMetrics
Metric Name:	UnauthorizedAccess
Metric Value:	1

$host $logName $user $timestamp $request $statusCode $size



Define an alarm with a specific threshold for that metric

Modify Alarm

Alarm Threshold

Provide the details and threshold for your alarm. Use the graph on the right to help set the appropriate threshold.

Name:	UnauthorizedAccess
Description:	
Whenever:	UnauthorizedAccess
is:	>= 5
for:	1 consecutive period(s)

Actions

Define what actions are taken when your alarm changes state.

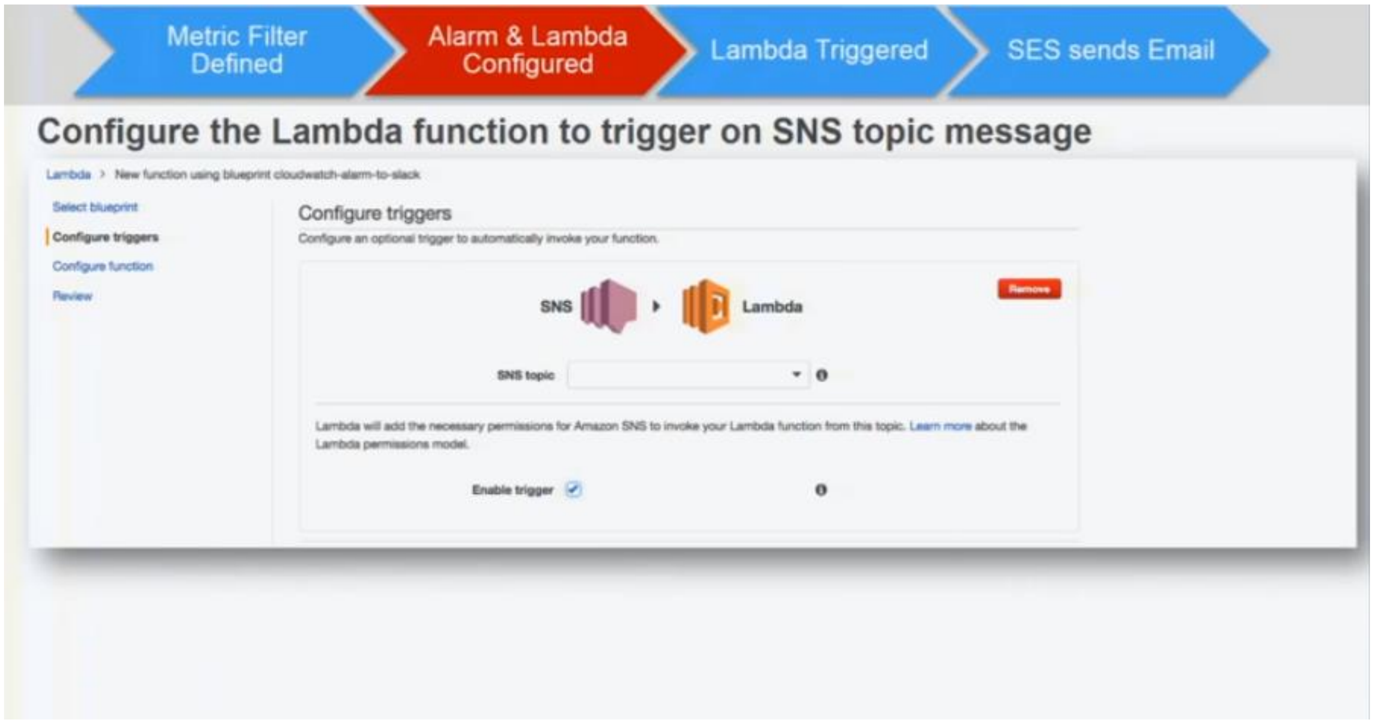
Notification

Whenever this alarm:	State is ALARM
Send notification to:	

Alarm Preview

This alarm will trigger when the blue line goes up to or above the red line for a duration of 5 minutes

Namespace:	LogMetrics
Metric Name:	UnauthorizedAccess
Period:	5 Minutes
Statistic:	Sum



Configure the Lambda function to trigger on SNS topic message

Lambda > New function using blueprint cloudwatch-alarm-to-slack

Select blueprint
Configure triggers
Configure function
Review

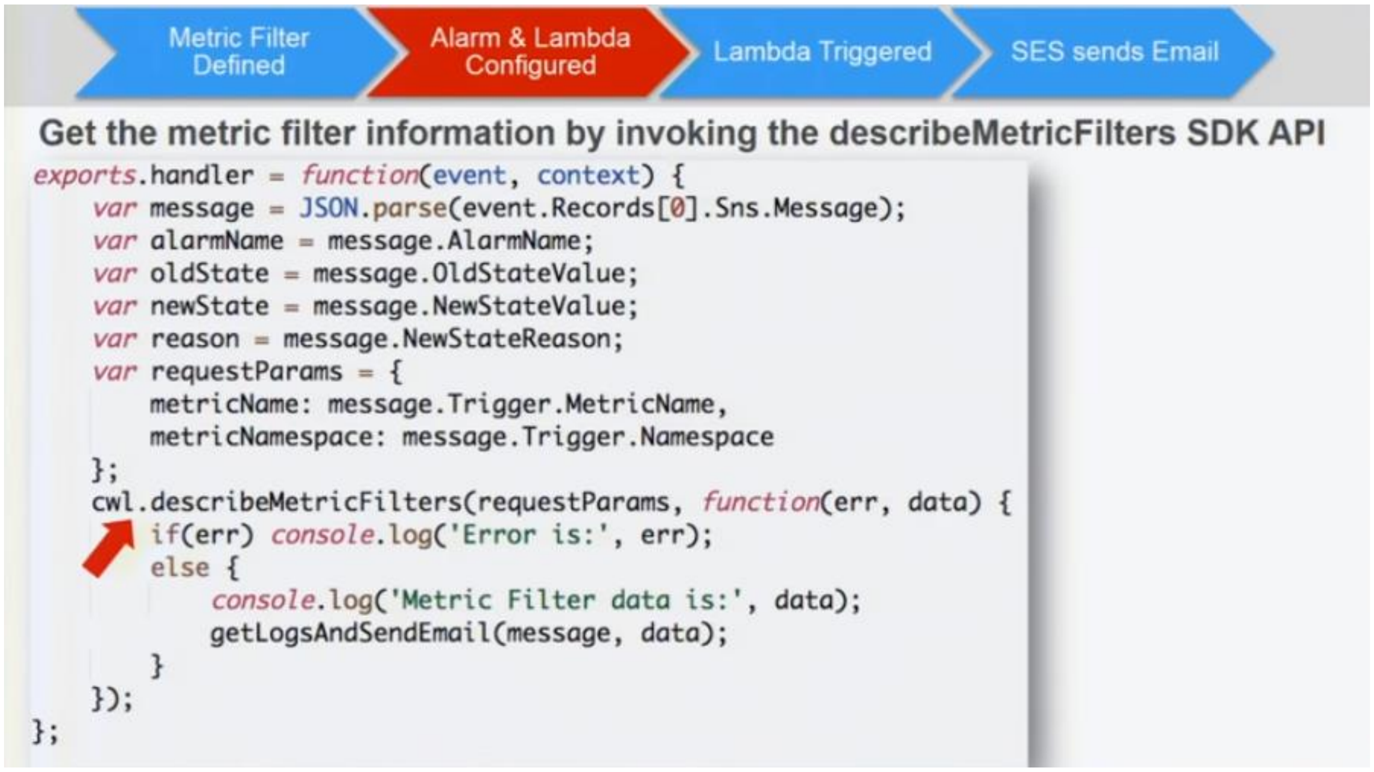
Configure triggers

Configure an optional trigger to automatically invoke your function.

SNS topic

Lambda will add the necessary permissions for Amazon SNS to invoke your Lambda function from this topic. Learn more about the Lambda permissions model.

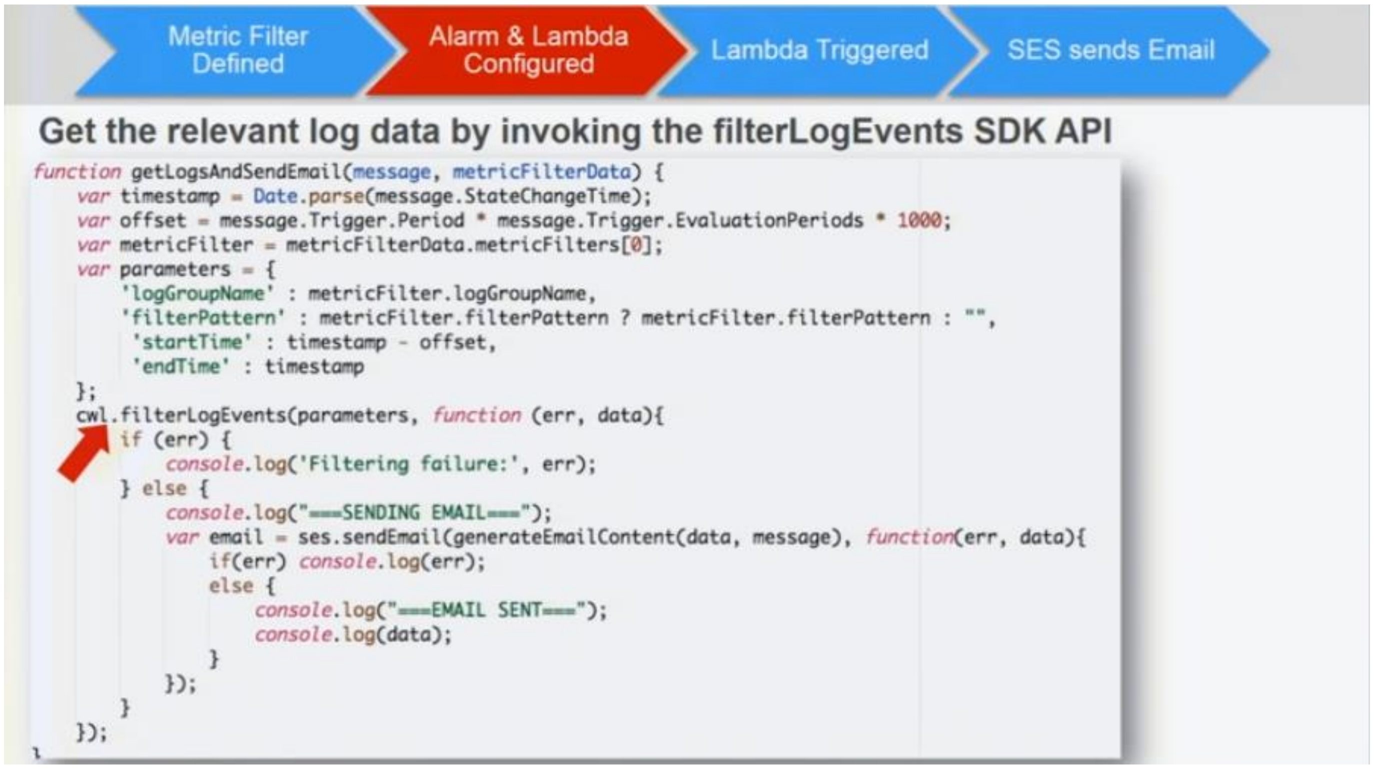
Enable trigger ✓



Get the metric filter information by invoking the describeMetricFilters SDK API

```
exports.handler = function(event, context) {
    var message = JSON.parse(event.Records[0].Sns.Message);
    var alarmName = message.AlarmName;
    var oldState = message.OldStateValue;
    var newState = message.NewStateValue;
    var reason = message.NewStateReason;
    var requestParams = {
        metricName: message.Trigger.MetricName,
        metricNamespace: message.Trigger.Namespace
    };
    cwl.describeMetricFilters(requestParams, function(err, data) {
        if(err) console.log('Error is:', err);
        else {
            console.log('Metric Filter data is:', data);
            getLogsAndSendEmail(message, data);
        }
    });
};
```

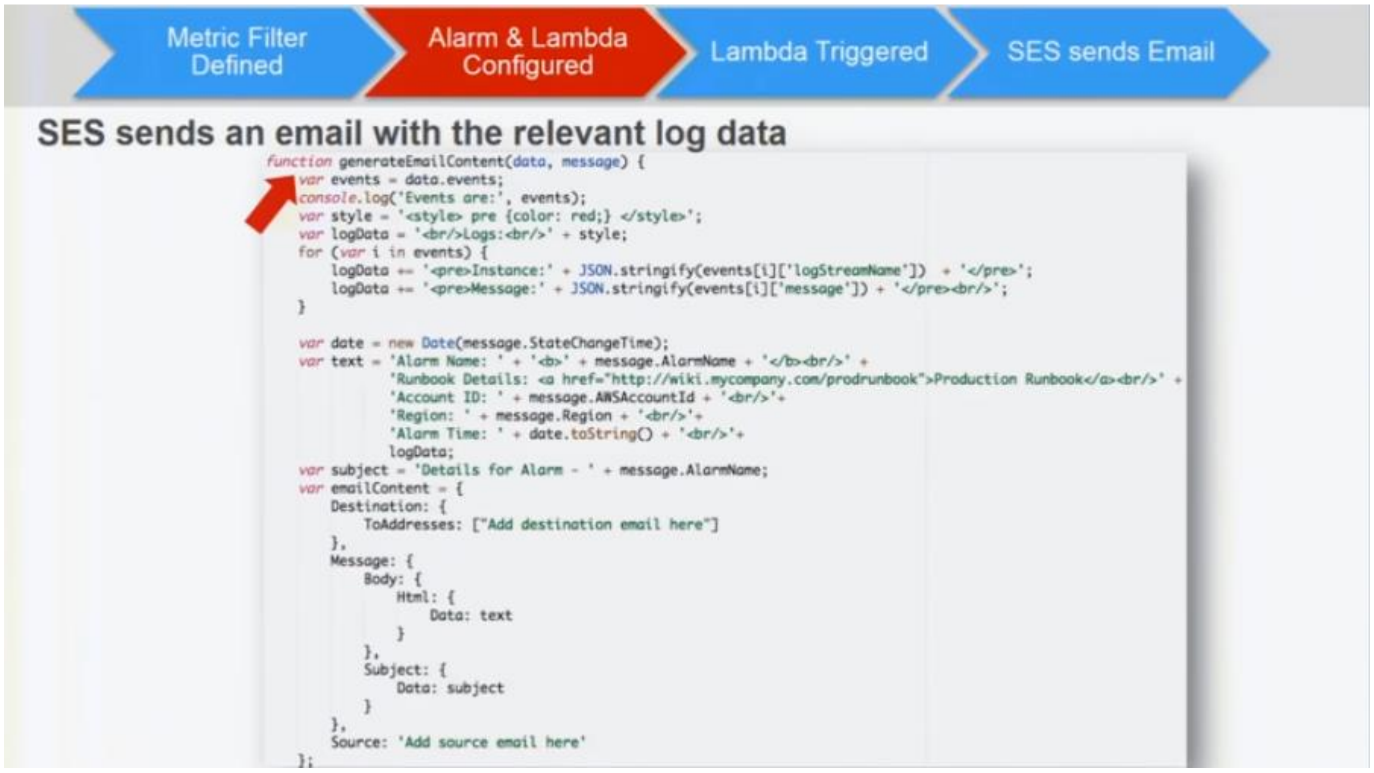
Configure the lambda as above that takes the alarm notification from the SNS message, pulls out some data like the MetricName and Namespace, then called the ***describeMetricFilters()*** public API using the AWS SDK



Get the relevant log data by invoking the filterLogEvents SDK API

```
function getLogsAndSendEmail(message, metricFilterData) {
    var timestamp = Date.parse(message.StateChangeTime);
    var offset = message.Trigger.Period * message.Trigger.EvaluationPeriods * 1000;
    var metricFilter = metricFilterData.metricFilters[0];
    var parameters = {
        'logGroupName' : metricFilter.logGroupName,
        'filterPattern' : metricFilter.filterPattern ? metricFilter.filterPattern : "",
         'startTime' : timestamp - offset,
         'endTime' : timestamp
    };
    cwl.filterLogEvents(parameters, function (err, data){
        if (err) {
            console.log('Filtering failure:', err);
        } else {
            console.log("===SENDING EMAIL===");
            var email = ses.sendEmail(generateEmailContent(data, message), function(err, data){
                if(err) console.log(err);
                else {
                    console.log("===EMAIL SENT===");
                    console.log(data);
                }
            });
        }
    });
}
```

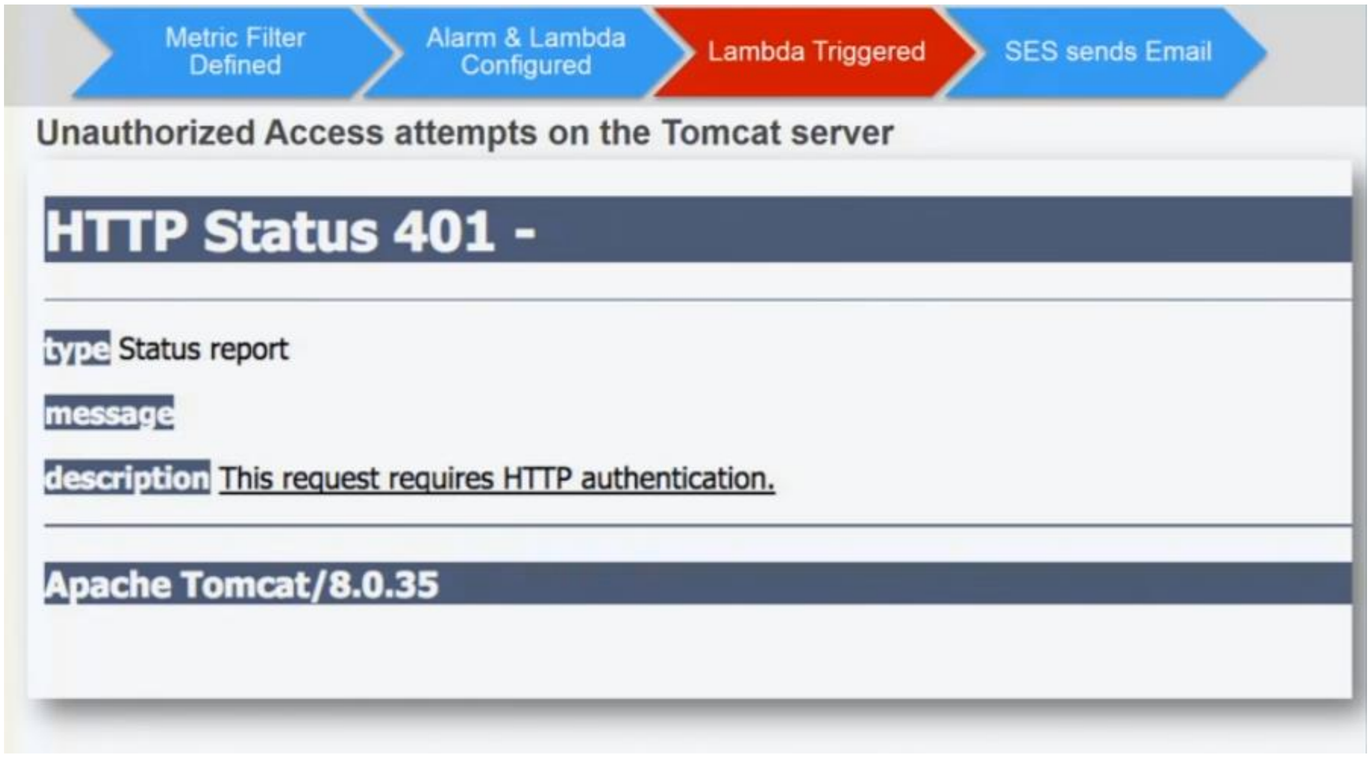
Then call the filterLogEvents() API with the parameters and data,



SES sends an email with the relevant log data

```
function generateEmailContent(data, message) {
    var events = data.events;
    console.log('Events are:', events);
    var style = '<style> pre {color: red;} </style>';
    var logData = '<br/>Logs:<br/>' + style;
    for (var i in events) {
        logData += '<pre>Instance:' + JSON.stringify(events[i]['logStreamName'])  + '</pre>';
        logData += '<pre>Message:' + JSON.stringify(events[i]['message']) + '</pre><br/>';
    }

    var date = new Date(message.StateChangeTime);
    var text = 'Alarm Name: ' + '<b>' + message.AlarmName + '</b><br/>' +
               'Runbook Details: <a href="http://wiki.mycompany.com/prodrunbook">Production Runbook</a><br/>' +
               'Account ID: ' + message.AWSAccountId + '<br/>'+
               'Region: ' + message.Region + '<br/>'+
               'Alarm Time: ' + date.toString() + '<br/>'+
               logData;
    var subject = 'Details for Alarm - ' + message.AlarmName;
    var emailContent = {
        Destination: {
            ToAddresses: ["Add destination email here"]
        },
        Message: {
            Body: {
                Html: {
                    Data: text
                }
            },
            Subject: {
                Data: subject
            }
        },
        Source: 'Add source email here'
    };
```

Send out the email with other details to the operator



Unauthorized Access attempts on the Tomcat server

HTTP Status 401 -

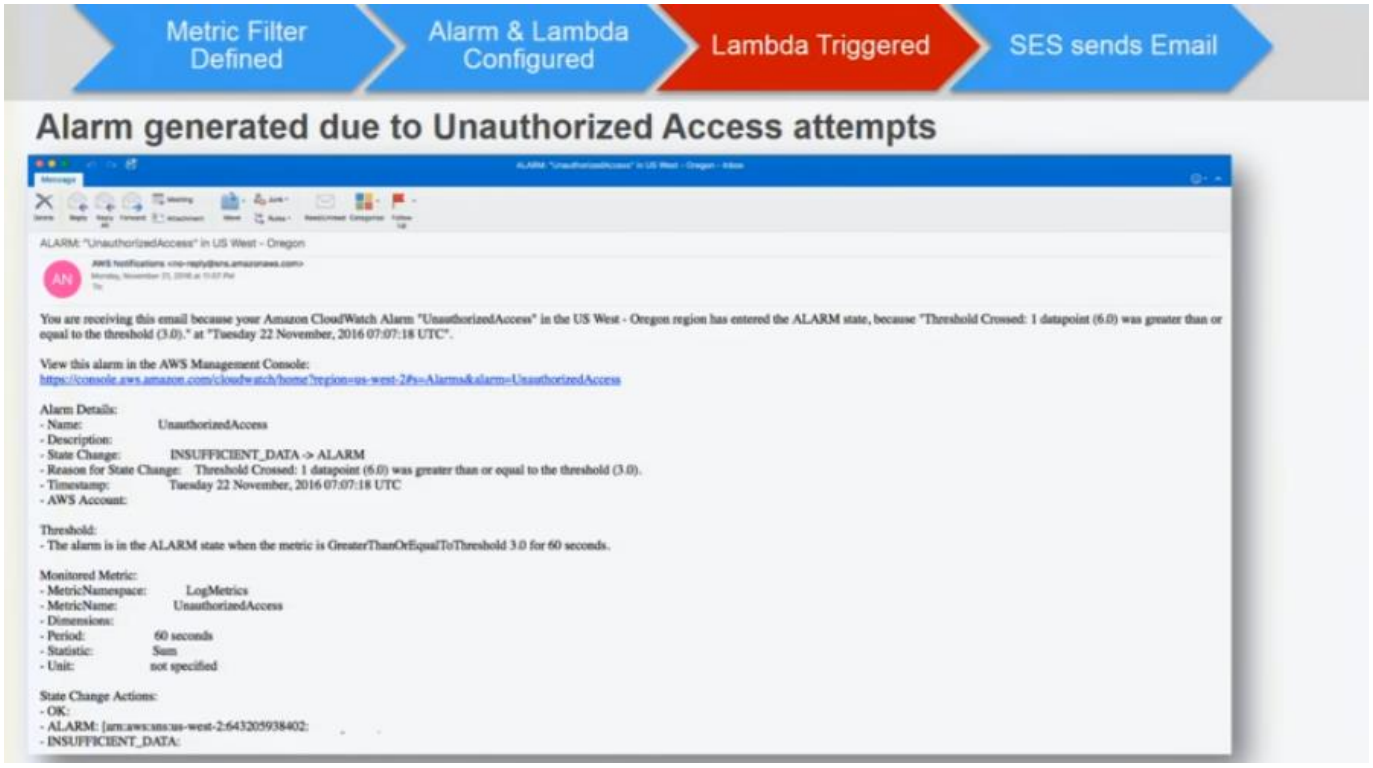
type Status report

message

description This request requires HTTP authentication.

Apache Tomcat/8.0.35

Test it out



Alarm generated due to Unauthorized Access attempts

ALARM: "UnauthorizedAccess" in US West - Oregon

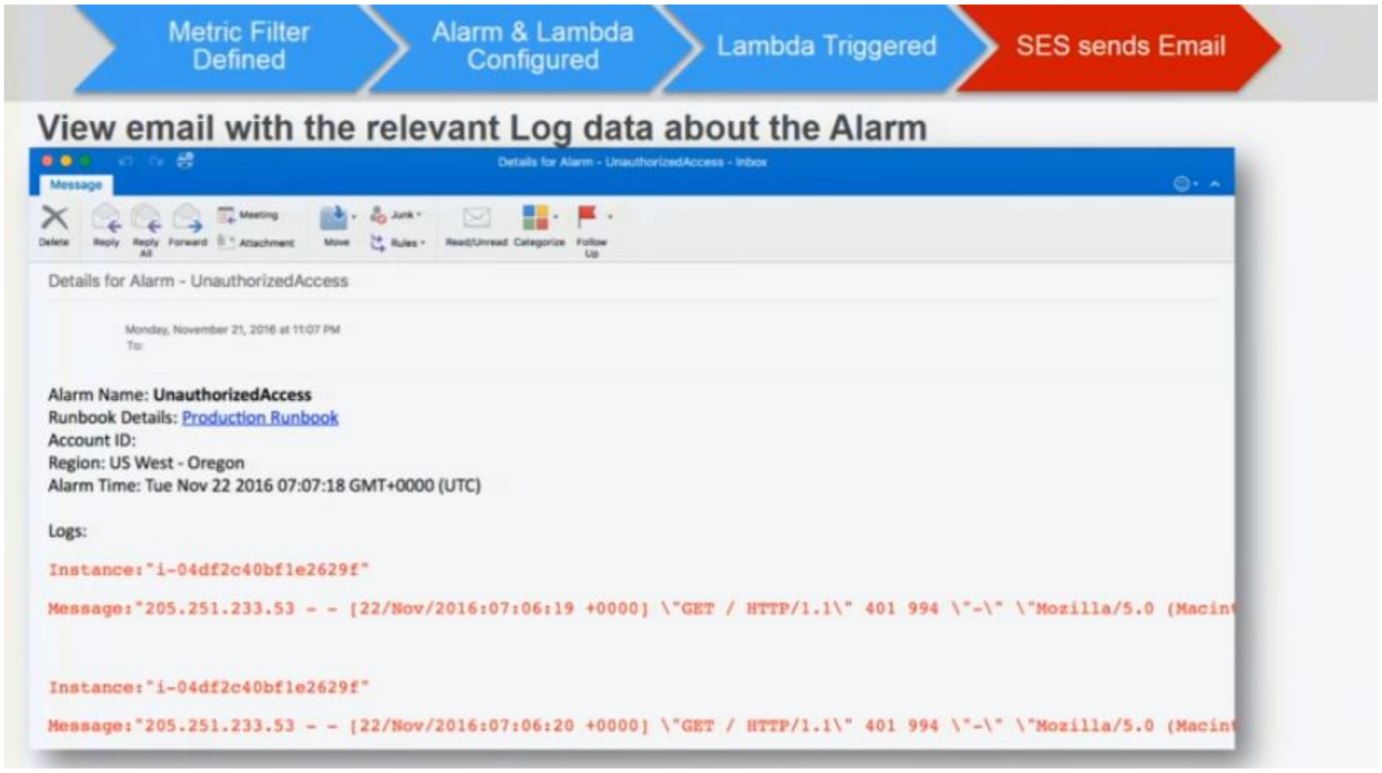
AWS Notifications <no-reply@sns.amazonaws.com>

You are receiving this email because your Amazon CloudWatch Alarm "UnauthorizedAccess" in the US West - Oregon region has entered the ALARM state, because "Threshold Crossed: 1 datapoint (6.0) was greater than or equal to the threshold (3.0)." at "Tuesday 22 November, 2016 07:07:18 UTC".

View this alarm in the AWS Management Console:
https://console.aws.amazon.com/cloudwatch/home?region=us-west-2#s=Alarms&alarm=UnauthorizedAccess

```
Alarm Details:
- Name:                    UnauthorizedAccess
- Description:
- State Change:            INSUFFICIENT_DATA -> ALARM
- Reason for State Change:    Threshold Crossed: 1 datapoint (6.0) was greater than or equal to the threshold (3.0).
- Timestamp:               Tuesday 22 November, 2016 07:07:18 UTC
- AWS Account:

Threshold:
- The alarm is in the ALARM state when the metric is GreaterThanOrEqualToThreshold 3.0 for 60 seconds.

Monitored Metric:
- MetricNamespace:         LogMetrics
- MetricName:              UnauthorizedAccess
- Dimensions:
- Period:                  60 seconds
- Statistic:               Sum
- Unit:                    not specified

State Change Actions:
- OK:
- ALARM: [arn:aws:sns:us-west-2:643205938402:
- INSUFFICIENT_DATA:
```

This is the alarm we will get



View email with the relevant Log data about the Alarm

Details for Alarm - UnauthorizedAccess

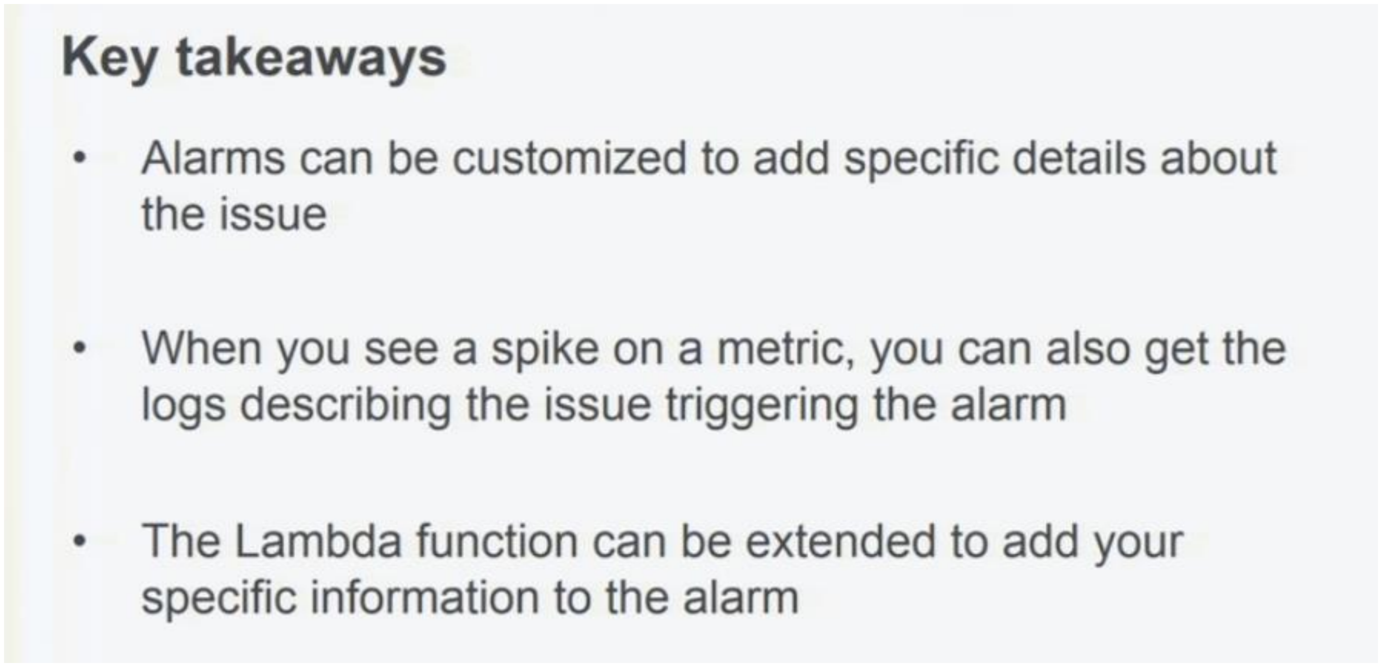
Monday, November 21, 2016 at 11:07 PM
To:

Alarm Name: **UnauthorizedAccess**
Runbook Details: Production Runbook
Account ID:
Region: US West - Oregon
Alarm Time: Tue Nov 22 2016 07:07:18 GMT+0000 (UTC)

Logs:

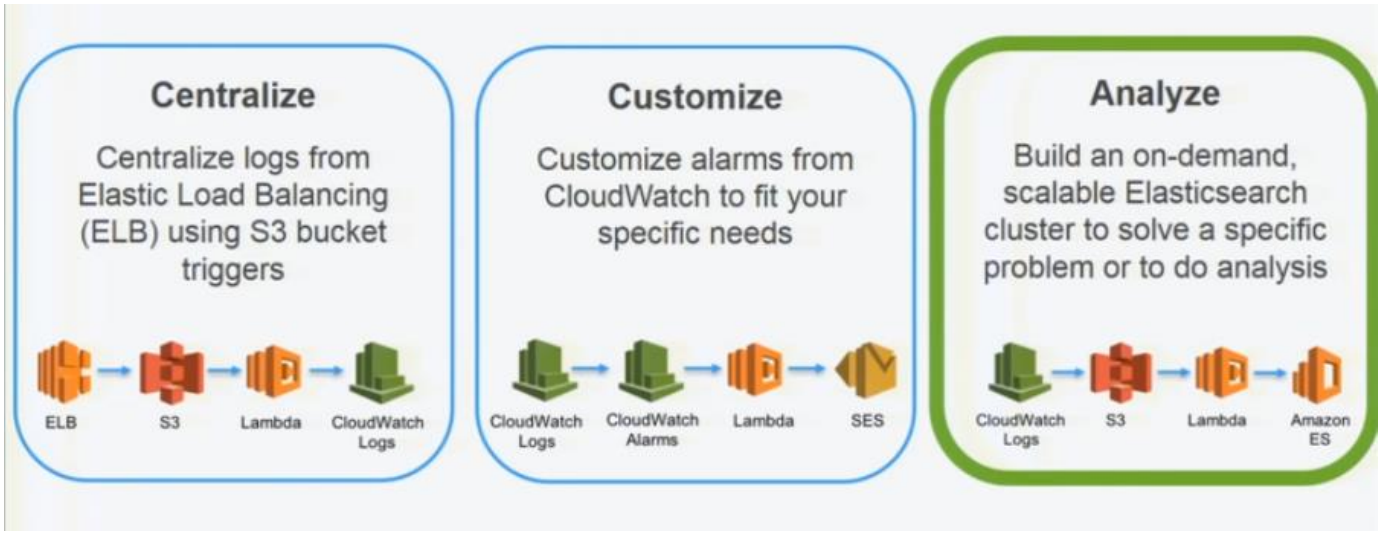
```
Instance:"i-04df2c40bf1e2629f"

Message:"205.251.233.53 - - [22/Nov/2016:07:06:19 +0000] \"GET / HTTP/1.1\" 401 994 \"-\" \"Mozilla/5.0 (Macint


Instance:"i-04df2c40bf1e2629f"

Message:"205.251.233.53 - - [22/Nov/2016:07:06:20 +0000] \"GET / HTTP/1.1\" 401 994 \"-\" \"Mozilla/5.0 (Macint
```

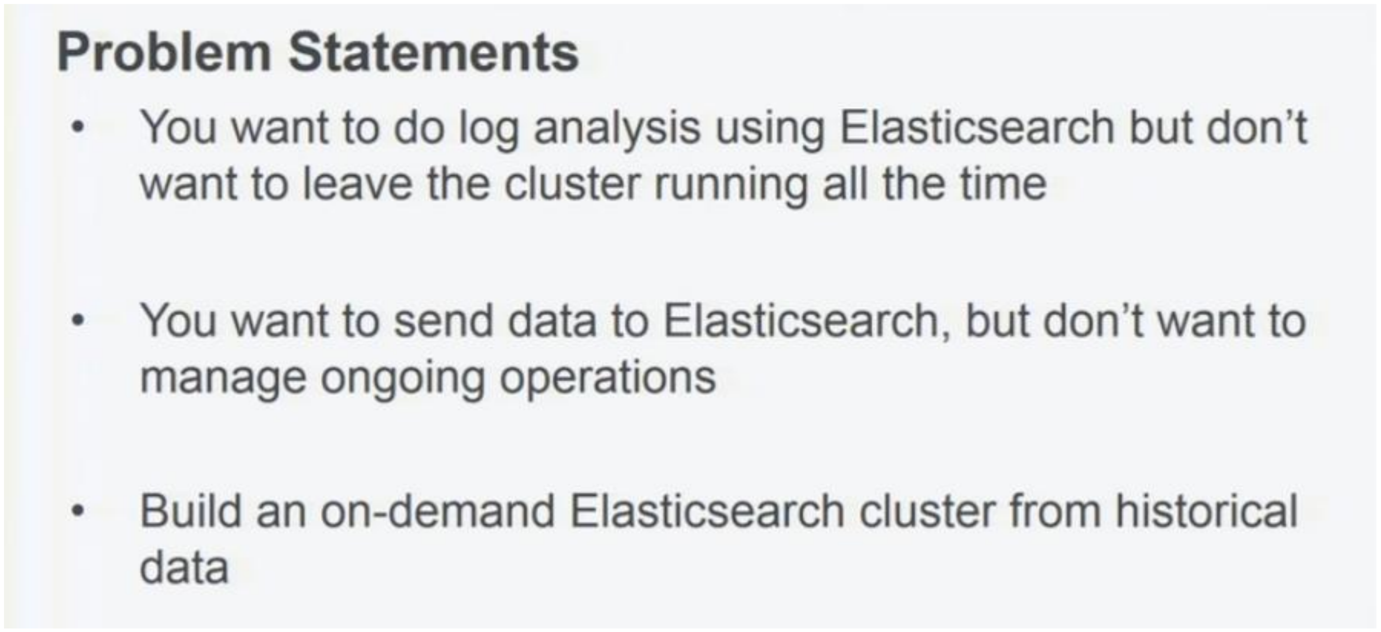
The email is much better with the details in it



Key takeaways

- Alarms can be customized to add specific details about the issue
- When you see a spike on a metric, you can also get the logs describing the issue triggering the alarm
- The Lambda function can be extended to add your specific information to the alarm


Centralize
Centralize logs from Elastic Load Balancing (ELB) using S3 bucket triggers
ELB
S3
Lambda
CloudWatch Logs
Customize
Customize alarms from CloudWatch to fit your specific needs
CloudWatch Logs
CloudWatch Alarms
Lambda
SES
Analyze
Build an on-demand, scalable Elasticsearch cluster to solve a specific problem or to do analysis
CloudWatch Logs
S3
Lambda
Amazon ES

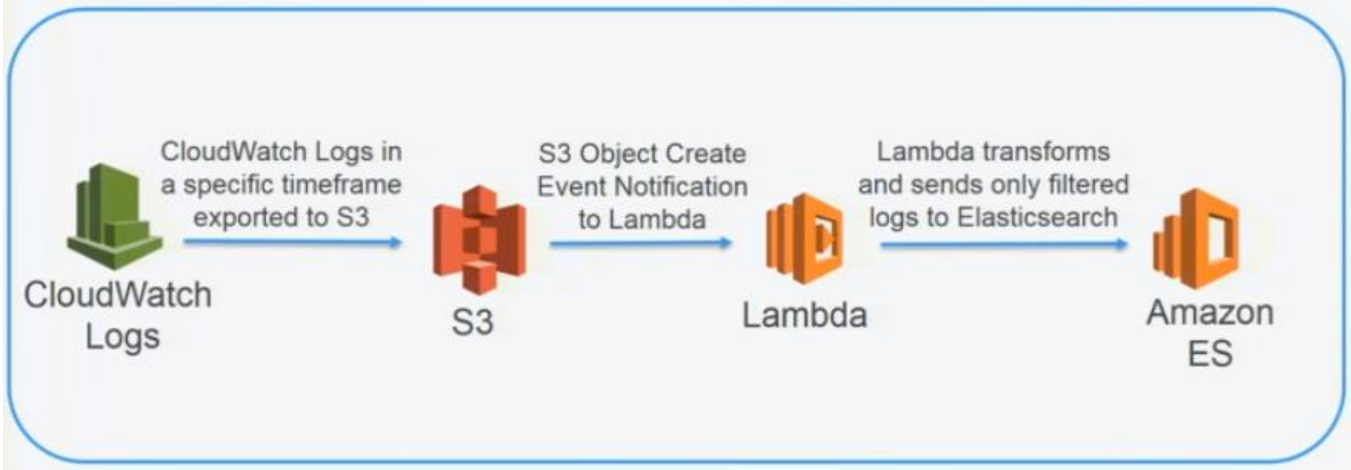


Problem Statements

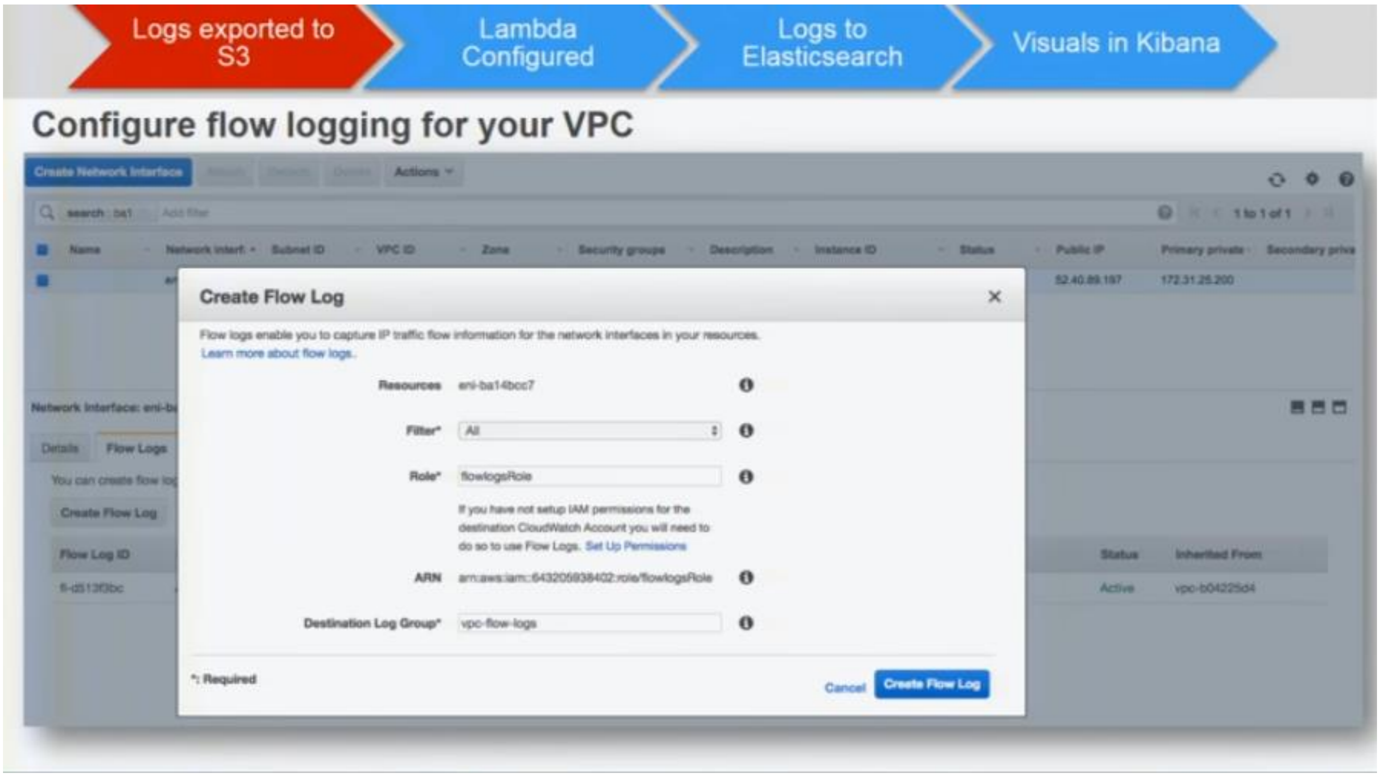
- You want to do log analysis using Elasticsearch but don't want to leave the cluster running all the time
- You want to send data to Elasticsearch, but don't want to manage ongoing operations
- Build an on-demand Elasticsearch cluster from historical data

You can integrate with the Amazon Elasticsearch service, data can flow into it in real time and then do real time analysis on it without leaving the lambda running everytime. We might want to accumulate the historical CW logs data and then do searching on it when needed by standing up an Elasticsearch cluster

Flow of Events


CloudWatch Logs in a specific timeframe exported to S3
S3 Object Create Event Notification to Lambda
Lambda transforms and sends only filtered logs to Elasticsearch
CloudWatch Logs
S3
Lambda
Amazon ES

You can also transform the logs by breaking them into smaller more indexable pieces before sending them into the ES endpoint.



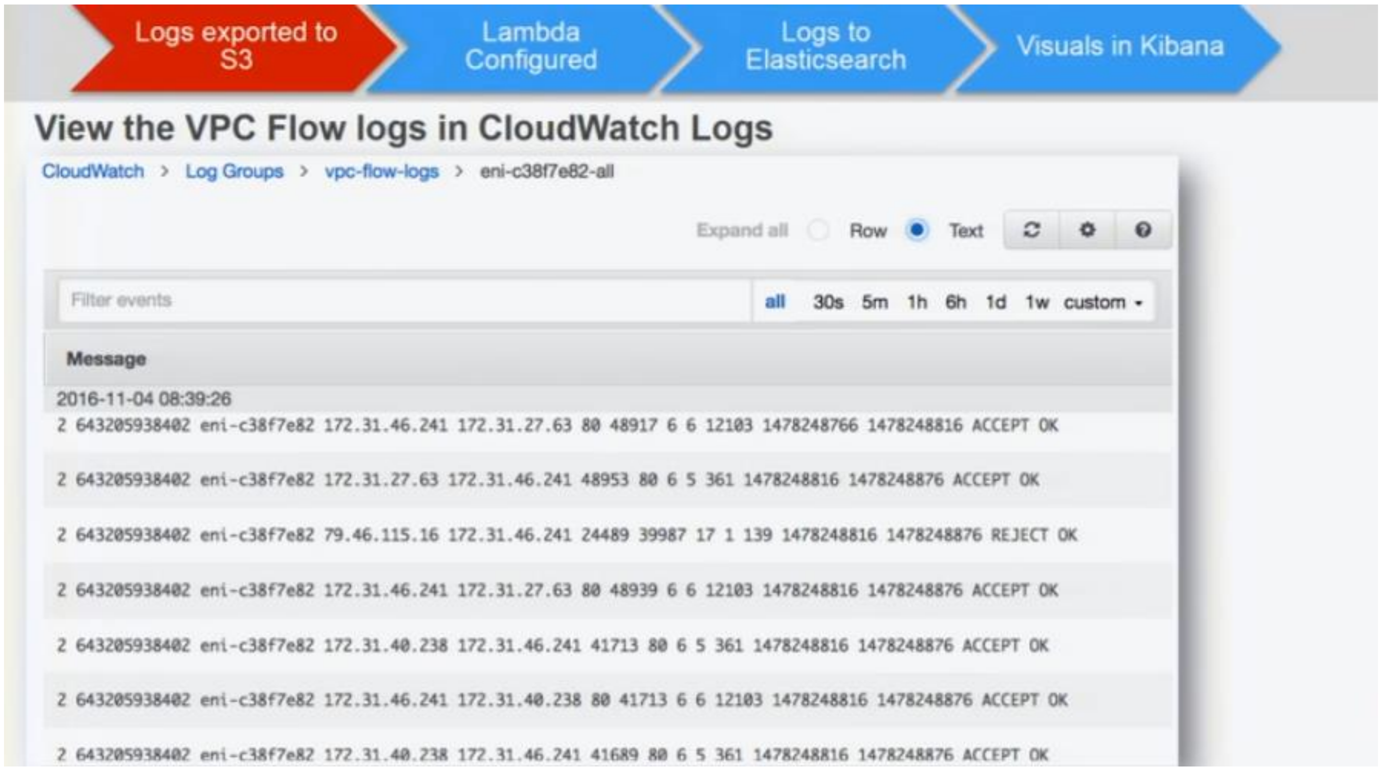
Configure flow logging for your VPC

Create Flow Log

Flow logs enable you to capture IP traffic flow information for the network interfaces in your resources.
Learn more about flow logs.

Resources	eni-ba14bcc7
Filter*	All
Role*	flowlogsRole
	If you have not setup IAM permissions for the destination CloudWatch Account you will need to do so to use Flow Logs. Set Up Permissions
ARN	arn:aws:iam::643205938402:role/flowlogsRole
Destination Log Group*	vpc-flow-logs

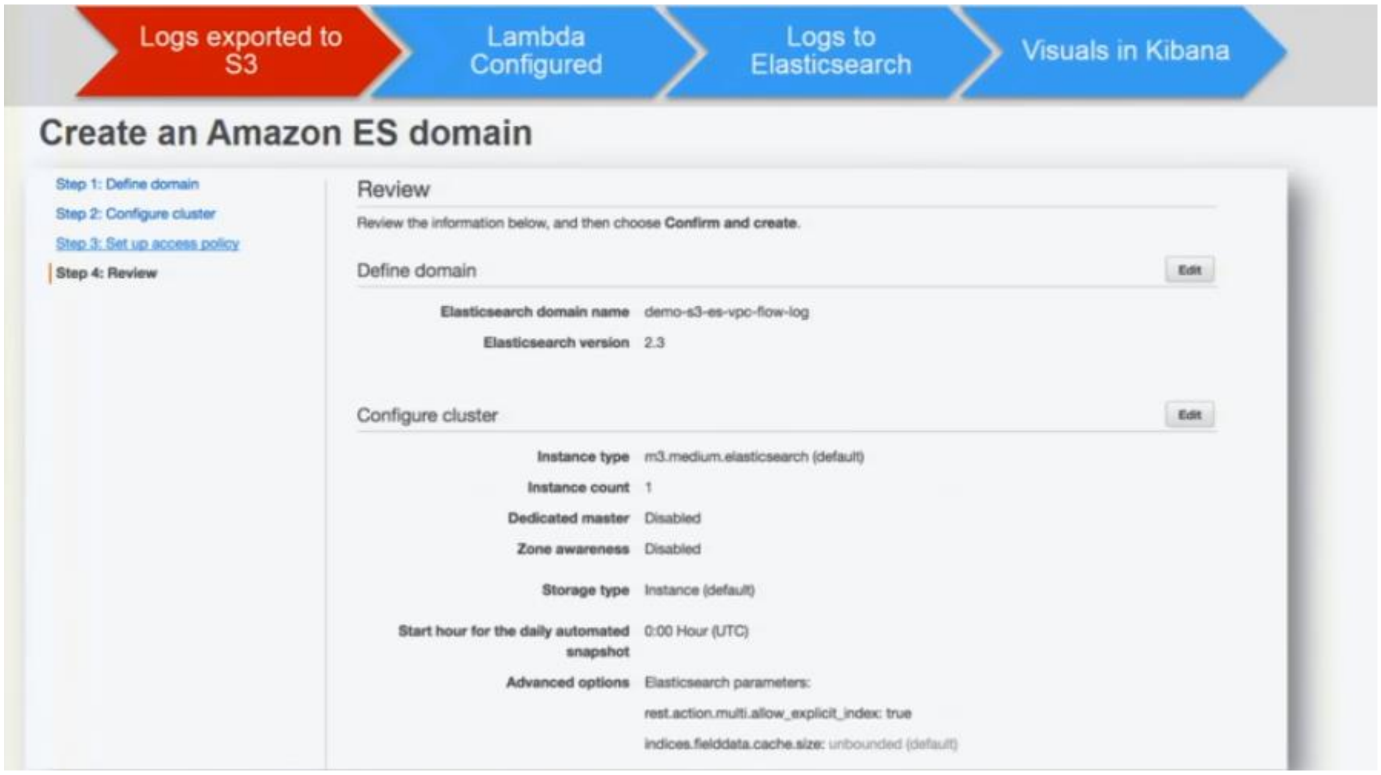
*: Required



View the VPC Flow logs in CloudWatch Logs

CloudWatch > Log Groups > vpc-flow-logs > eni-c38f7e82-all
Expand all
Row
Text
Filter events
all 30s 5m 1h 6h 1d 1w custom

```
Message
2016-11-04 08:39:26
2 643205938402 eni-c38f7e82 172.31.46.241 172.31.27.63 80 48917 6 6 12103 1478248766 1478248816 ACCEPT OK

2 643205938402 eni-c38f7e82 172.31.27.63 172.31.46.241 48953 80 6 5 361 1478248816 1478248876 ACCEPT OK

2 643205938402 eni-c38f7e82 79.46.115.16 172.31.46.241 24489 39987 17 1 139 1478248816 1478248876 REJECT OK

2 643205938402 eni-c38f7e82 172.31.46.241 172.31.27.63 80 48939 6 6 12103 1478248816 1478248876 ACCEPT OK

2 643205938402 eni-c38f7e82 172.31.40.238 172.31.46.241 41713 80 6 5 361 1478248816 1478248876 ACCEPT OK

2 643205938402 eni-c38f7e82 172.31.46.241 172.31.40.238 80 41713 6 6 12103 1478248816 1478248876 ACCEPT OK

2 643205938402 eni-c38f7e82 172.31.40.238 172.31.46.241 41689 80 6 5 361 1478248816 1478248876 ACCEPT OK
```



Create an Amazon ES domain

Review

Review the information below, and then choose **Confirm and create**.

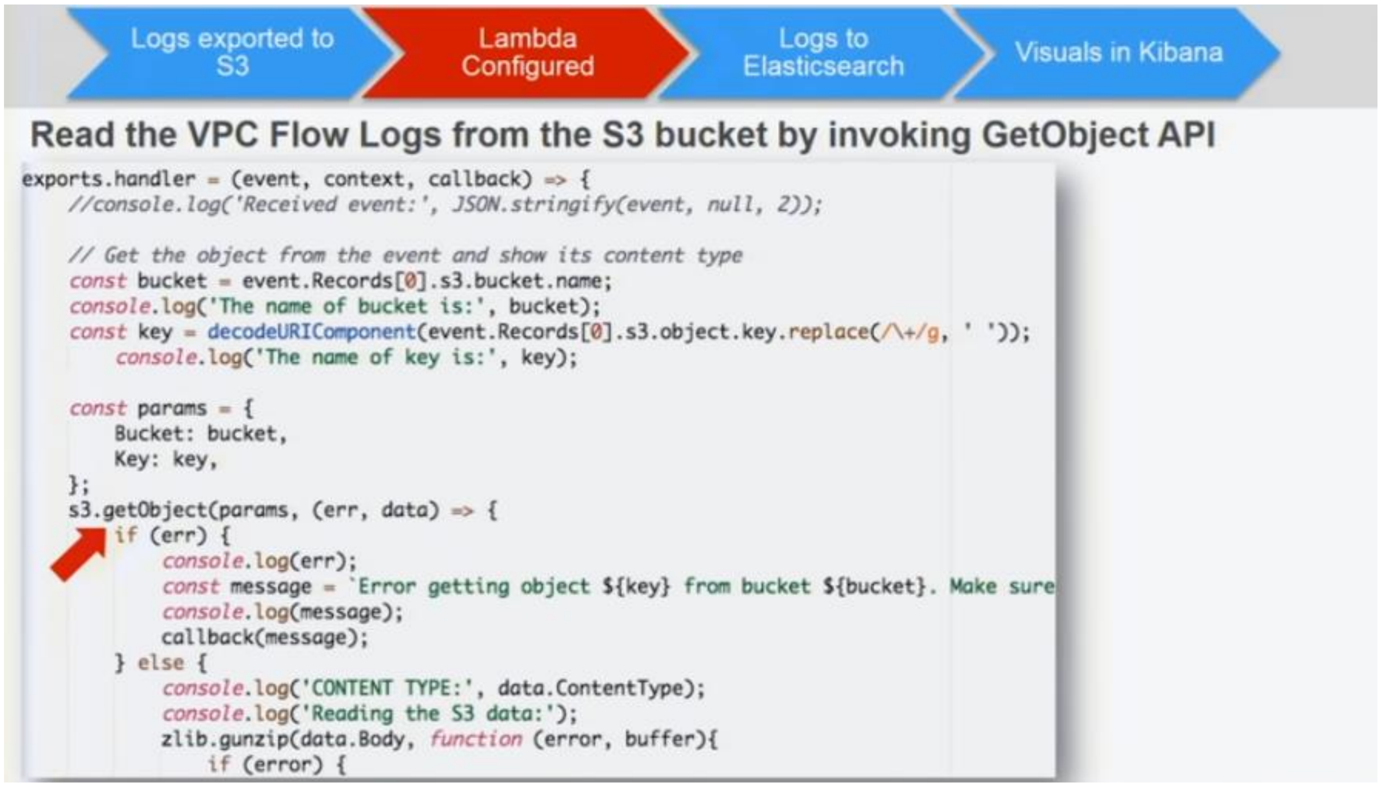
Define domain

Elasticsearch domain name	demo-s3-es-vpc-flow-log
Elasticsearch version	2.3

Configure cluster

Instance type	m3.medium.elasticsearch (default)
Instance count	1
Dedicated master	Disabled
Zone awareness	Disabled
Storage type	Instance (default)
Start hour for the daily automated snapshot	0:00 Hour (UTC)
Advanced options	Elasticsearch parameters:
	rest.action.multi.allow_explicit_index: true
	indices.fielddata.cache.size: unbounded (default)

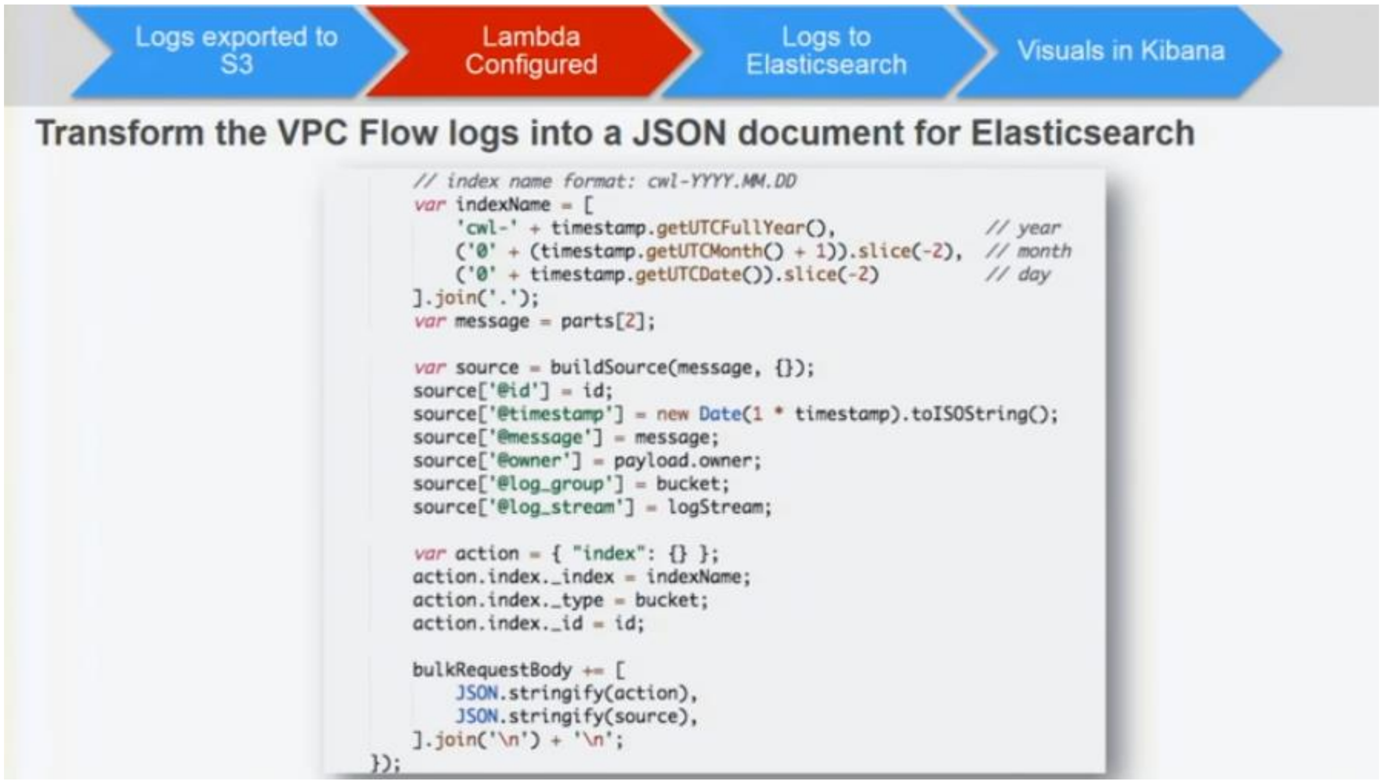
We create an ES domain, set it up with the number of nodes that we need, the types of data, etc



Read the VPC Flow Logs from the S3 bucket by invoking GetObject API

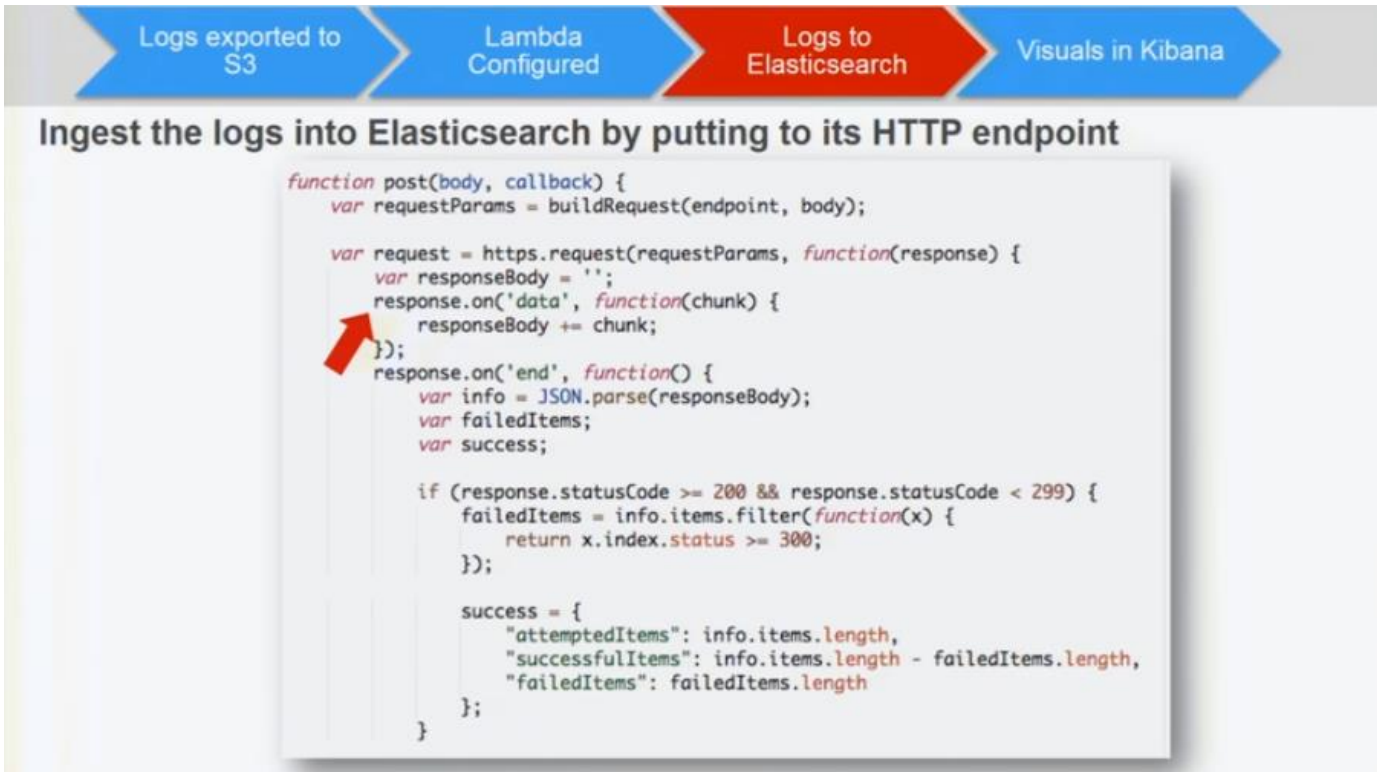
```
exports.handler = (event, context, callback) => {
    //console.log('Received event:', JSON.stringify(event, null, 2));

    // Get the object from the event and show its content type
    const bucket = event.Records[0].s3.bucket.name;
    console.log('The name of bucket is:', bucket);
    const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, ' '));
        console.log('The name of key is:', key);

    const params = {
        Bucket: bucket,
        Key: key,
    };
    s3.getObject(params, (err, data) => {
        if (err) {
            console.log(err);
            const message = `Error getting object ${key} from bucket ${bucket}. Make sure
            console.log(message);
            callback(message);
        } else {
            console.log('CONTENT TYPE:', data.ContentType);
            console.log('Reading the S3 data:');
            zlib.gunzip(data.Body, function (error, buffer){
                if (error) {
```

We also set up our lambda function to trigger on the ***s3ObjectCreate()*** event to send the data to ES



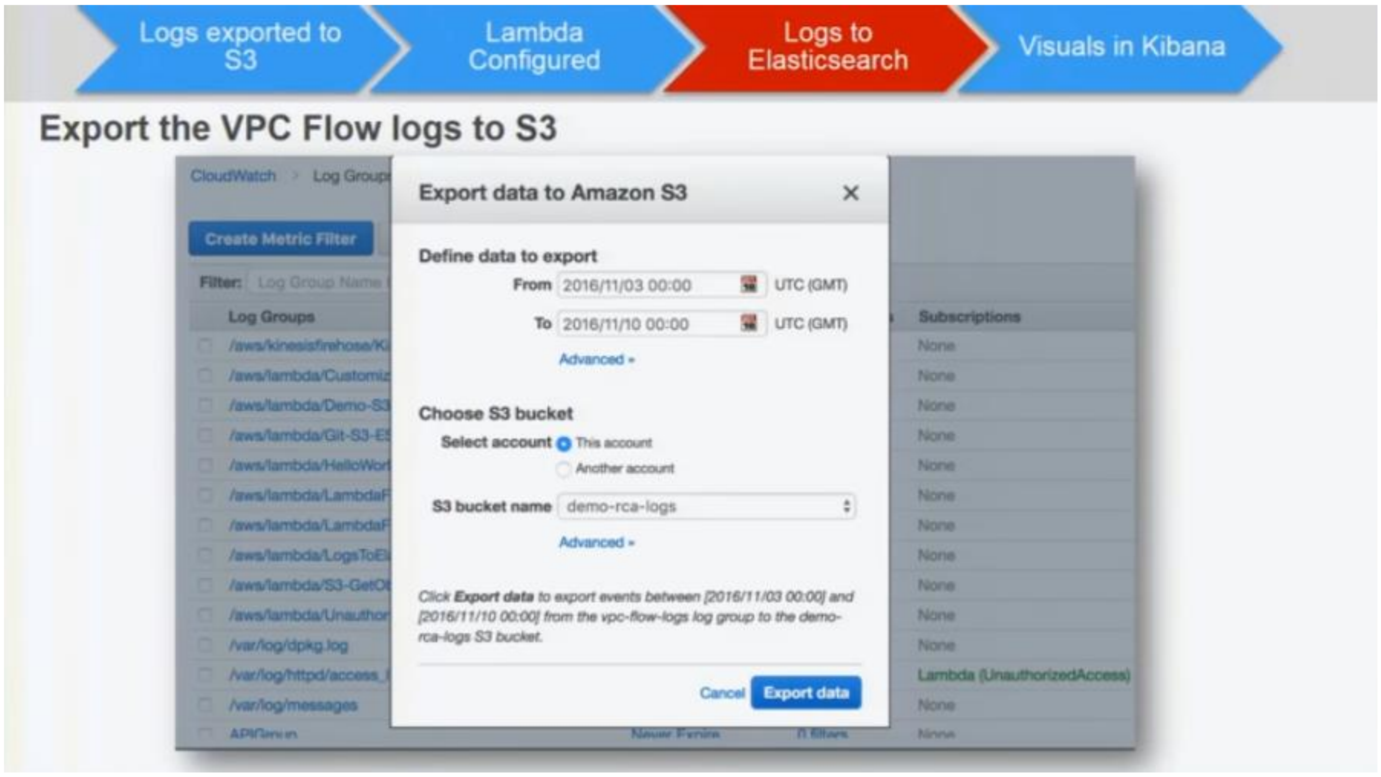
Transform the VPC Flow logs into a JSON document for Elasticsearch

```
        // index name format: cwl-YYYY.MM.DD
        var indexName = [
            'cwl-' + timestamp.getUTCFullYear(),           // year
            ('0' + (timestamp.getUTCMonth() + 1)).slice(-2),  // month
            ('0' + timestamp.getUTCDate()).slice(-2)          // day
        ].join('.');
        var message = parts[2];

        var source = buildSource(message, {});
        source['@id'] = id;
        source['@timestamp'] = new Date(1 * timestamp).toISOString();
        source['@message'] = message;
        source['@owner'] = payload.owner;
        source['@log_group'] = bucket;
        source['@log_stream'] = logStream;

        var action = { "index": {} };
        action.index._index = indexName;
        action.index._type = bucket;
        action.index._id = id;

        bulkRequestBody += [
            JSON.stringify(action),
            JSON.stringify(source),
        ].join('\n') + '\n';
    });
```


Logs exported to S3
Lambda Configured
Logs to Elasticsearch
Visuals in Kibana

Ingest the logs into Elasticsearch by putting to its HTTP endpoint

```
function post(body, callback) {
    var requestParams = buildRequest(endpoint, body);

    var request = https.request(requestParams, function(response) {
        var responseBody = '';
        response.on('data', function(chunk) {
            responseBody += chunk;
        });
        response.on('end', function() {
            var info = JSON.parse(responseBody);
            var failedItems;
            var success;

            if (response.statusCode >= 200 && response.statusCode < 299) {
                failedItems = info.items.filter(function(x) {
                    return x.index.status >= 300;
                });

                success = {
                    "attemptedItems": info.items.length,
                    "successfulItems": info.items.length - failedItems.length,
                    "failedItems": failedItems.length
                };
            }
```



Export the VPC Flow logs to S3

Log Groups	Subscriptions
/aws/kinesisfirehose/K	None
/aws/lambda/Customiz	None
/aws/lambda/Demo-S3	None
/aws/lambda/Git-S3-E	None
/aws/lambda/HelloWor	None
/aws/lambda/LambdaF	None
/aws/lambda/LambdaF	None
/aws/lambda/LogsToEl	None
/aws/lambda/S3-GetO	None
/aws/lambda/Unauthor	None
/var/log/dpkg.log	None
/var/log/httpd/access_l	Lambda (UnauthorizedAccess)
/var/log/messages	None
APIGroup	None

Export data to Amazon S3

Define data to export

From: 2016/11/03 00:00 UTC (GMT)

To: 2016/11/10 00:00 UTC (GMT)

Advanced »

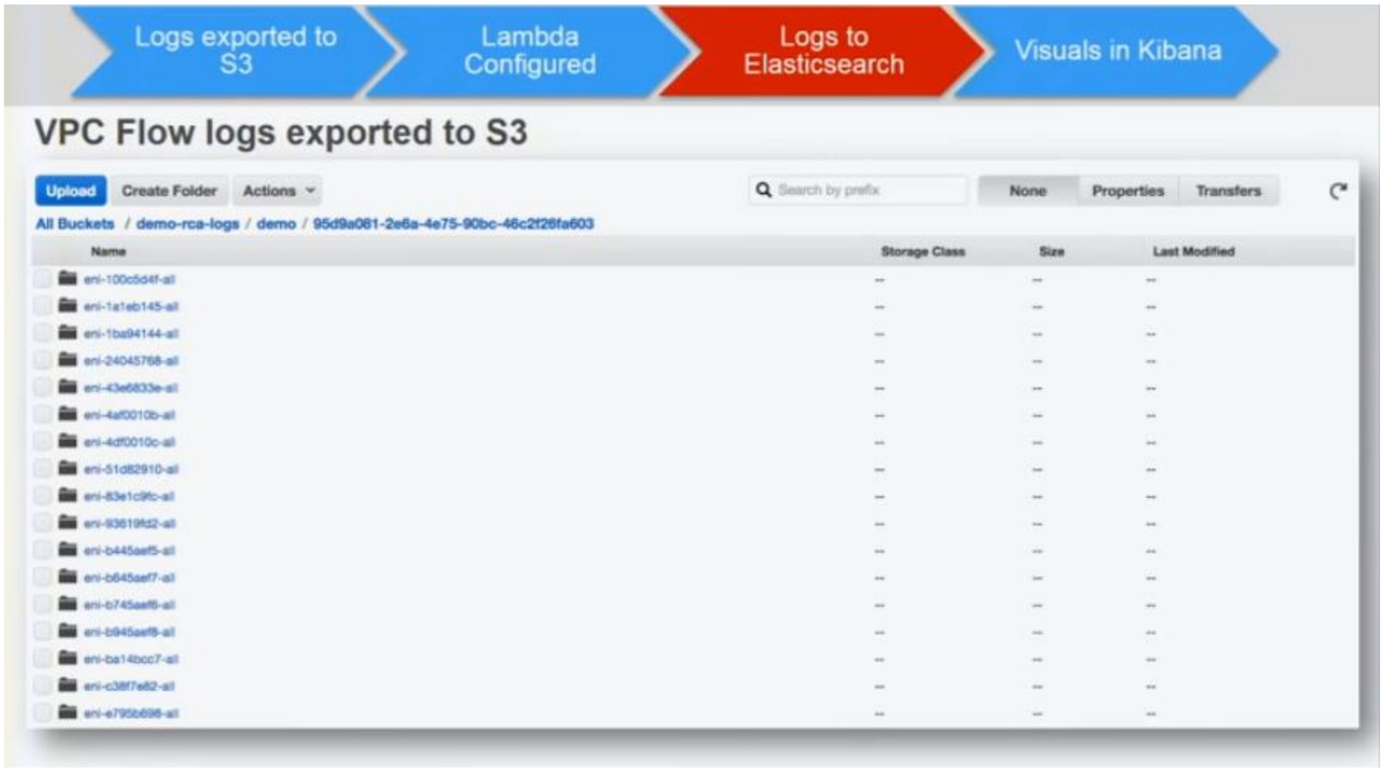
Choose S3 bucket

Select account: [✓ This account] Another account

S3 bucket name: demo-rca-logs

Advanced »

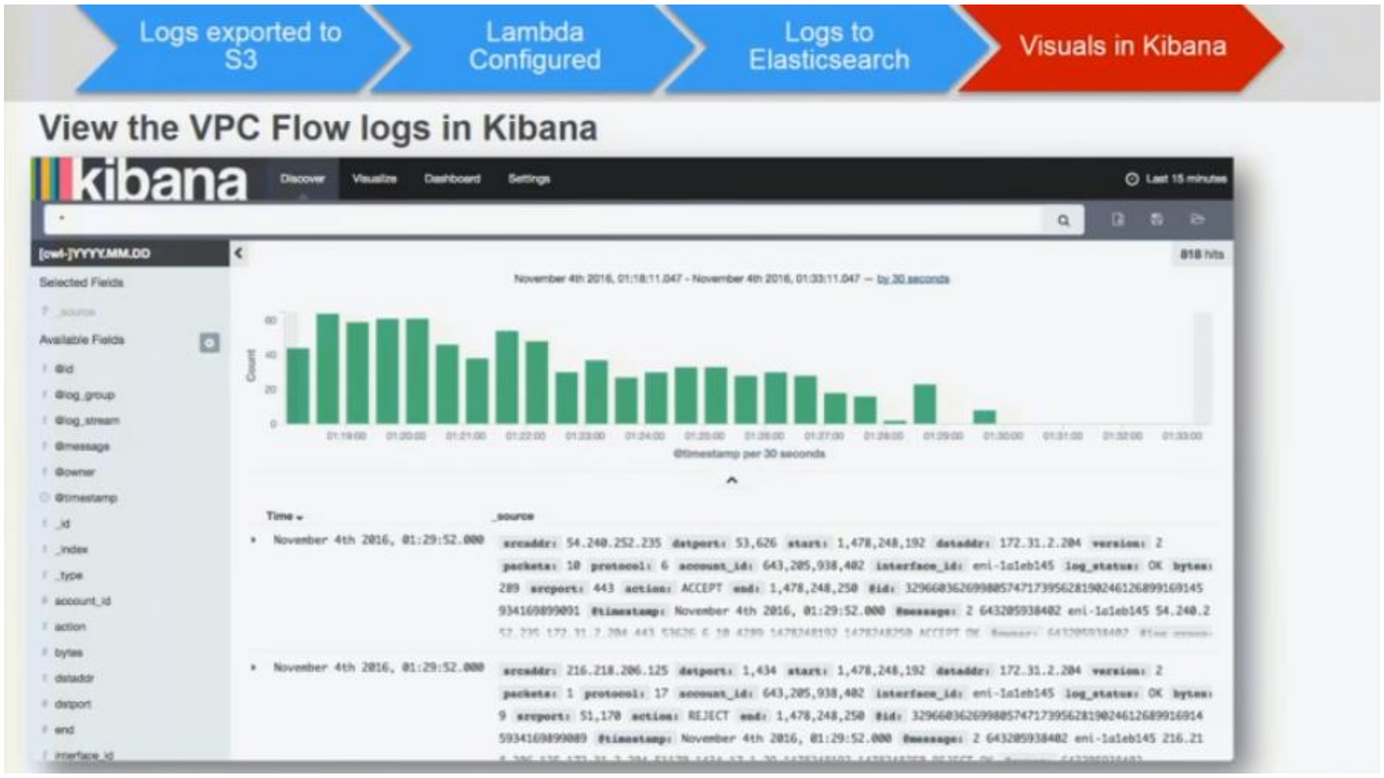
*Click **Export data** to export events between [2016/11/03 00:00] and [2016/11/10 00:00] from the vpc-flow-logs log group to the demo-rca-logs S3 bucket.*



VPC Flow logs exported to S3

All Buckets / demo-rca-logs / demo / 95d9a061-2e6a-4e75-90bc-46c2f26fa603

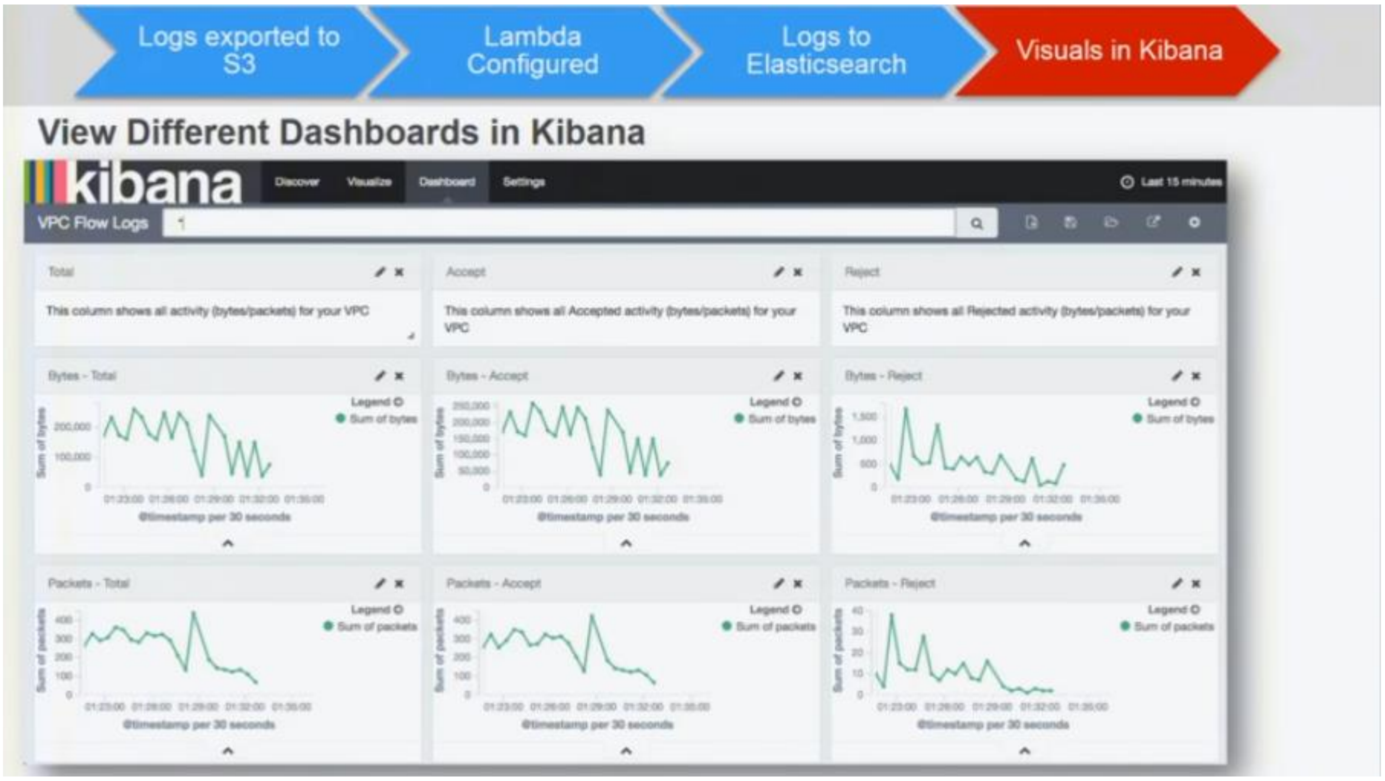
Name	Storage Class	Size	Last Modified
eni-100c5d4f-all	--	--	--
eni-1a1eb145-all	--	--	--
eni-1ba94144-all	--	--	--
eni-24045768-all	--	--	--
eni-43e6833e-all	--	--	--
eni-4af0010b-all	--	--	--
eni-4df0010c-all	--	--	--
eni-51d82910-all	--	--	--
eni-83e1c9fc-all	--	--	--
eni-93619fd2-all	--	--	--
eni-b445aef5-all	--	--	--
eni-b645aef7-all	--	--	--
eni-b745aef6-all	--	--	--
eni-b945aef8-all	--	--	--
eni-ba14bcc7-all	--	--	--
eni-c38f7e82-all	--	--	--
eni-e795b698-all	--	--	--



View the VPC Flow logs in Kibana

November 4th 2016, 01:18:11.047 - November 4th 2016, 01:33:11.047 — by 30 seconds

```
Time ▾                              _source
▸ November 4th 2016, 01:29:52.000  srcaddr: 54.240.252.235 dstport: 53,626 start: 1,478,248,192 dstaddr: 172.31.2.204 version: 2
                                   packets: 10 protocol: 6 account_id: 643,205,938,402 interface_id: eni-1a1eb145 log_status: OK bytes:
                                   289 srcport: 443 action: ACCEPT end: 1,478,248,250 @id: 329660362699805747173956281902461268991691 45
                                   934169899091 @timestamp: November 4th 2016, 01:29:52.000 @message: 2 643205938402 eni-1a1eb145 54.240.2
                                   52.235 172.31.2.204 443 53626 6 10 4289 1478248192 1478248250 ACCEPT OK @owner: 643205938402 @log_group:

▸ November 4th 2016, 01:29:52.000  srcaddr: 216.218.206.125 dstport: 1,434 start: 1,478,248,192 dstaddr: 172.31.2.204 version: 2
                                   packets: 1 protocol: 17 account_id: 643,205,938,402 interface_id: eni-1a1eb145 log_status: OK bytes:
                                   9 srcport: 51,170 action: REJECT end: 1,478,248,250 @id: 3296603626998057471739562819024612689916914
                                   5934169899089 @timestamp: November 4th 2016, 01:29:52.000 @message: 2 643205938402 eni-1a1eb145 216.21
```



View Different Dashboards in Kibana

Total

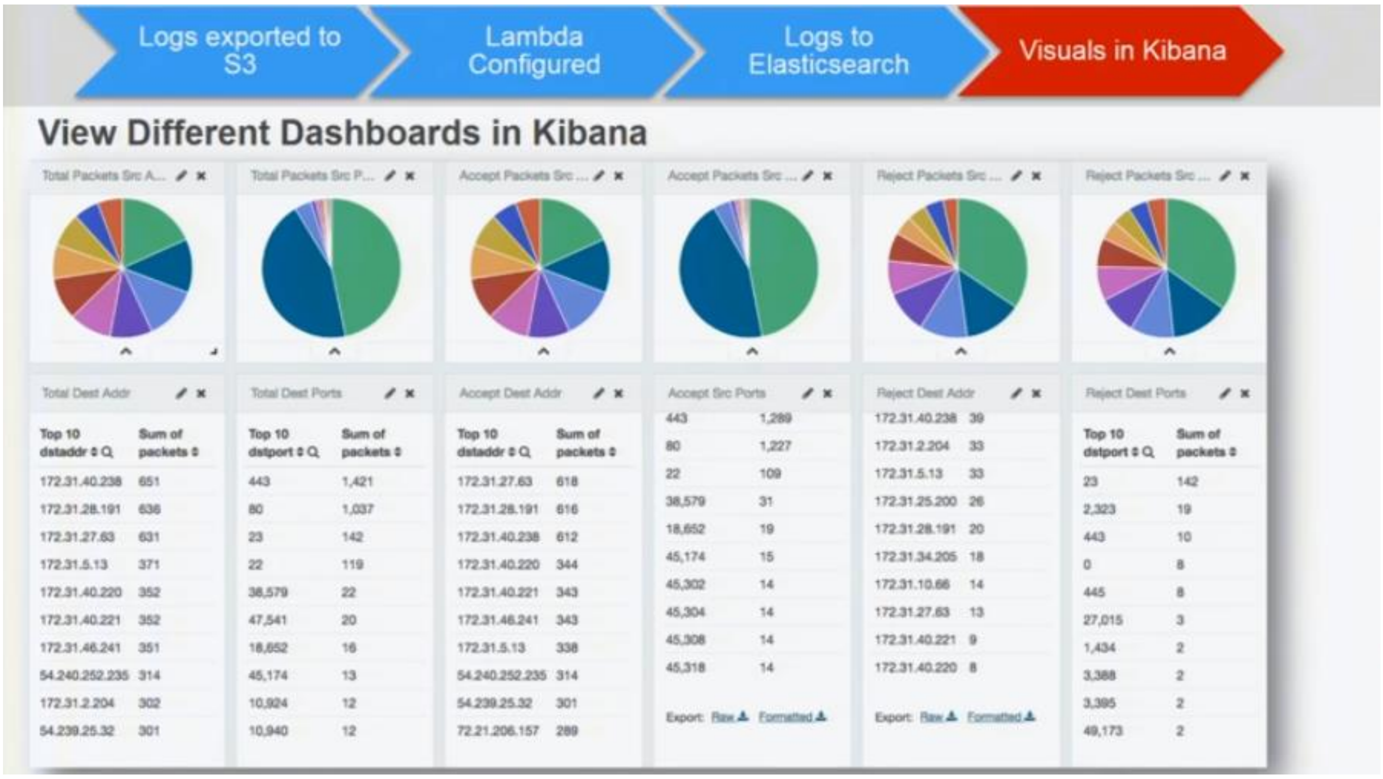
This column shows all activity (bytes/packets) for your VPC

Accept

This column shows all Accepted activity (bytes/packets) for your VPC

Reject

This column shows all Rejected activity (bytes/packets) for your VPC



View Different Dashboards in Kibana

Total Dest Addr

Top 10 dstaddr	Sum of packets
172.31.40.238	651
172.31.28.191	636
172.31.27.63	631
172.31.5.13	371
172.31.40.220	352
172.31.40.221	352
172.31.46.241	351
54.240.252.235	314
172.31.2.204	302
54.239.25.32	301

Total Dest Ports

Top 10 dstport	Sum of packets
443	1,421
80	1,037
23	142
22	119
38,579	22
47,541	20
18,652	16
45,174	13
10,924	12
10,940	12

Accept Dest Addr

Top 10 dstaddr	Sum of packets
172.31.27.63	618
172.31.28.191	616
172.31.40.238	612
172.31.40.220	344
172.31.40.221	343
172.31.46.241	343
172.31.5.13	338
54.240.252.235	314
54.239.25.32	301
72.21.206.157	289

Accept Src Ports

443	1,289
80	1,227
22	109
38,579	31
18,652	19
45,174	15
45,302	14
45,304	14
45,308	14
45,318	14

Export: Raw Formatted

Reject Dest Addr

172.31.40.238	39
172.31.2.204	33
172.31.5.13	33
172.31.25.200	26
172.31.28.191	20
172.31.34.205	18
172.31.10.66	14
172.31.27.63	13
172.31.40.221	9
172.31.40.220	8

Export: Raw Formatted

Reject Dest Ports

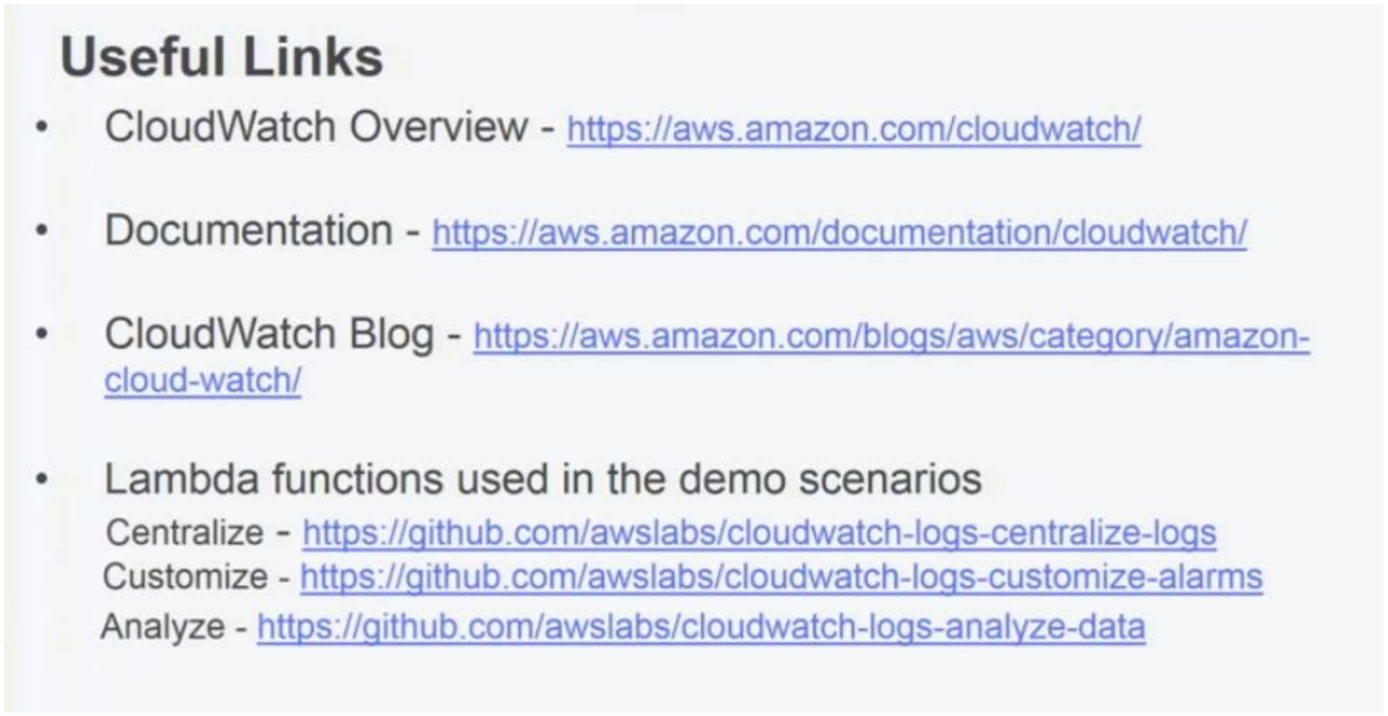
Top 10 dstport	Sum of packets
23	142
2,323	19
443	10
0	8
445	8
27,015	3
1,434	2
3,388	2
3,395	2
49,173	2

Key takeaways

- Send historical data within a timeframe to Elasticsearch on demand
- This reduces cost, burden of scalability, and operations time
- Troubleshooting gets easier because you have only limited and relevant data

Recap

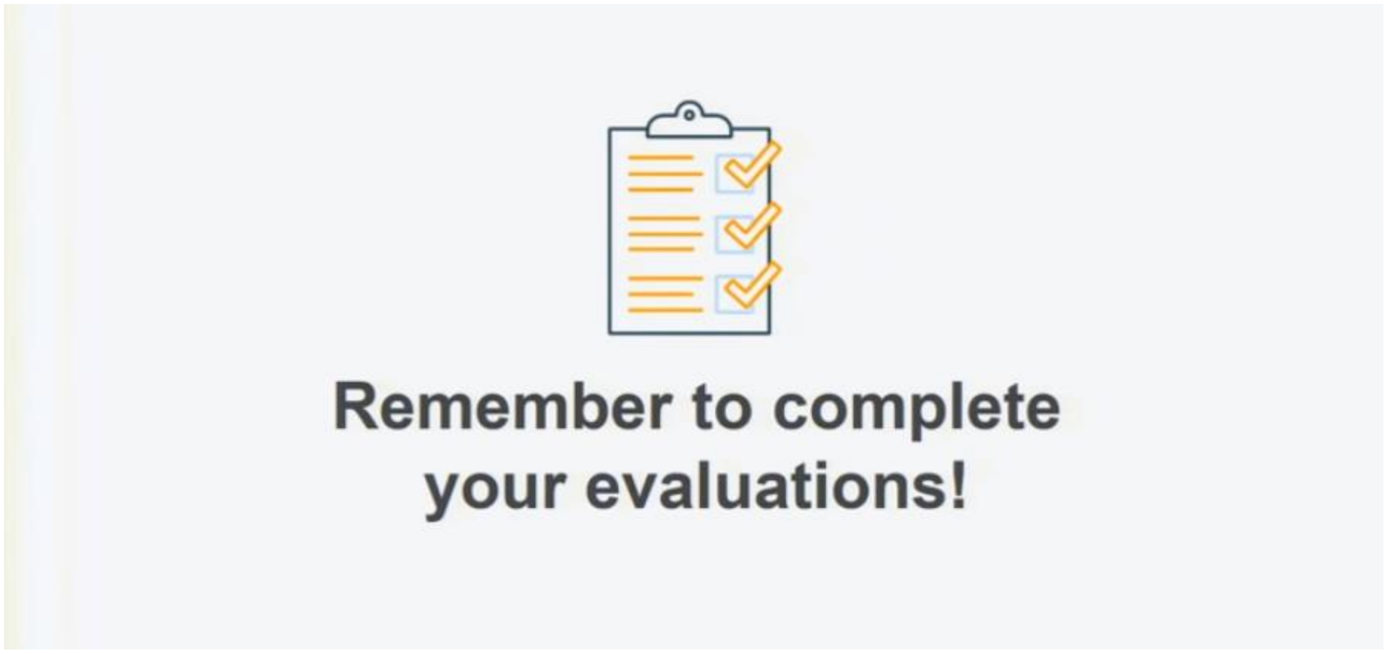
- Monitoring is more important than ever, but still too hard
- CloudWatch is working to make monitoring easier
- CloudWatch Logs and Lambda are powerful tools to tailor your monitoring for your business needs



Useful Links

- CloudWatch Overview - https://aws.amazon.com/cloudwatch/
- Documentation - https://aws.amazon.com/documentation/cloudwatch/
- CloudWatch Blog - https://aws.amazon.com/blogs/aws/category/amazon-cloud-watch/
- Lambda functions used in the demo scenarios

  Centralize - https://github.com/awslabs/cloudwatch-logs-centralize-logs
  Customize - https://github.com/awslabs/cloudwatch-logs-customize-alarms
  Analyze - https://github.com/awslabs/cloudwatch-logs-analyze-data



Remember to complete your evaluations!