SRV319

# How Nextdoor Built a Scalable, Serverless Data Pipeline for Billions of Events per Day

Prakash Janakiraman · Co-Founder & Chief Architect
Matt Wise · Sr. Systems Architect
Slava Markeyev · Software Engineer

Nextdoor

November 29, 2017

AWS re:Invent



aws

# Speakers

Nextdoor

Prakash Janakiraman

Co-Founder & Chief Architect

Matt Wise

Sr. Systems Architect

Slava Markeyev

Systems Engineer

# What to Expect from the Session

Nextdoor

- About Nextdoor
- Overview of our legacy pipeline
- Requirements for new pipeline
- Recap of AWS Serverless offerings
- Tool we built to help standardize Serverless ETL
- Deployment example via Cloudformation
- Q&A

Nextdoor

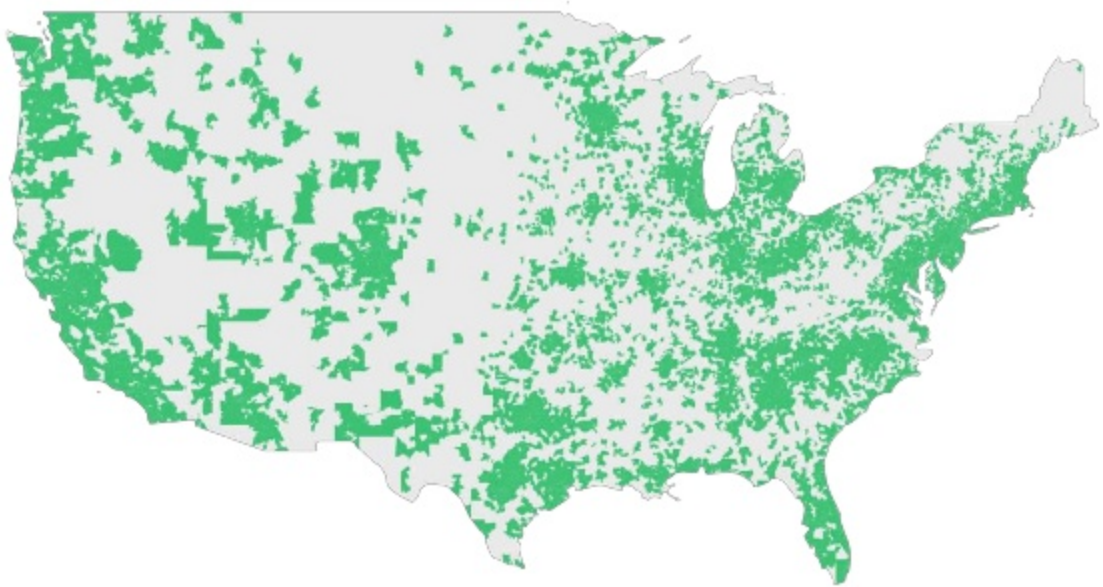

AWS re:Invent



aws

# Used by 80% of US neighborhoods

Nextdoor







aws

# Data drives products at Nextdoor

Nextdoor



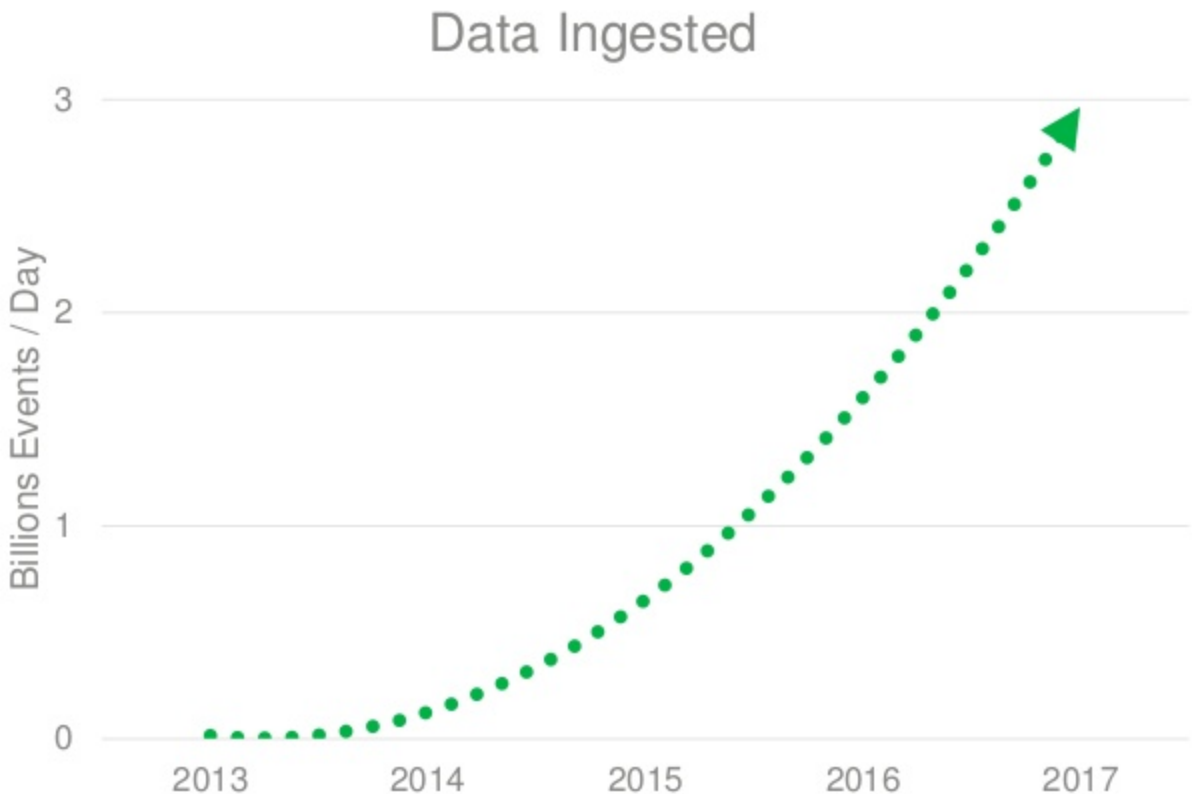




aws

# Data at Nextdoor

Nextdoor







aws

# Realtime Data Ingestion Pipeline

Slava Markeyev · Software Engineer

Nextdoor

aws re:Invent



aws

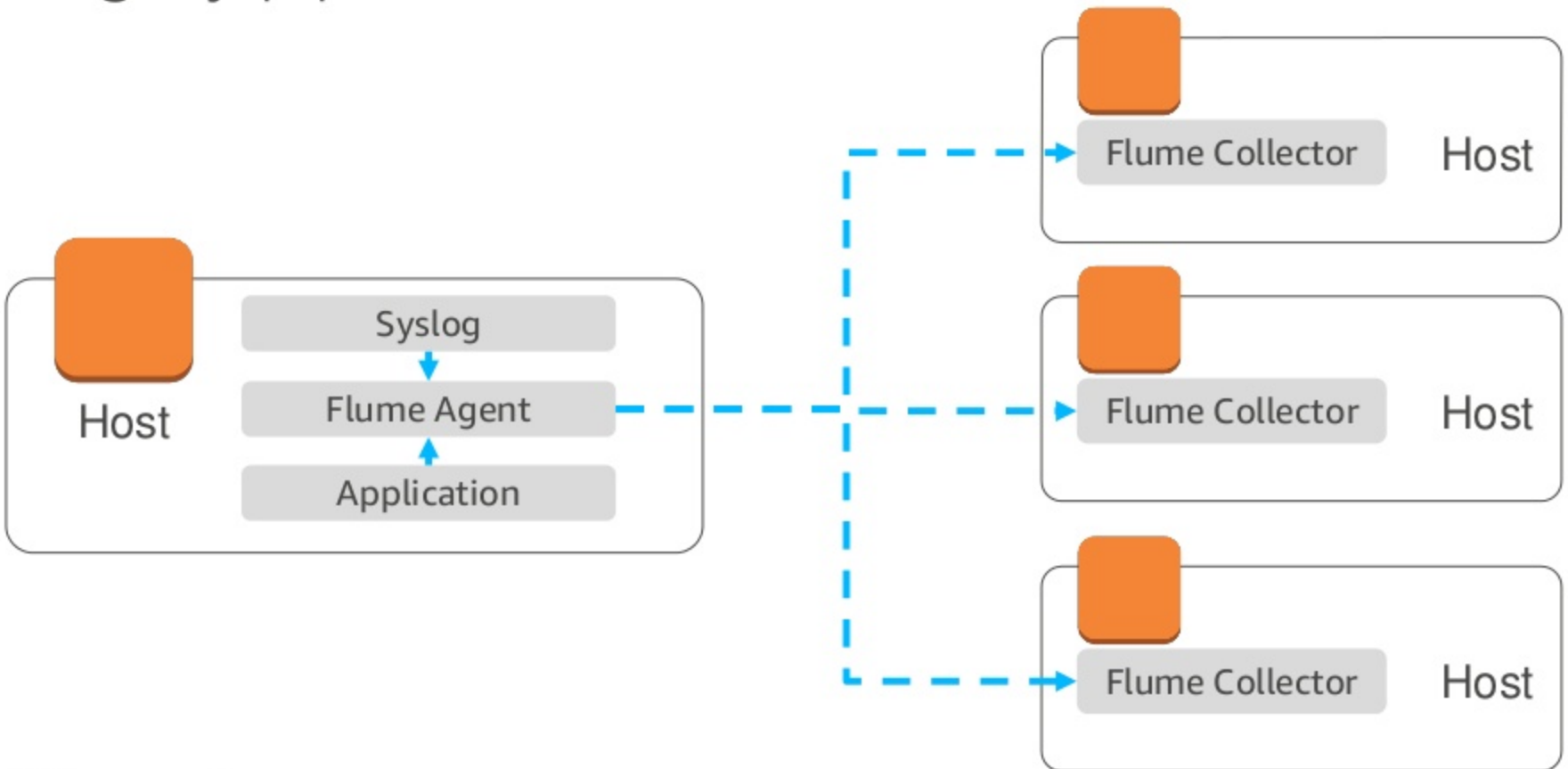
# Legacy pipeline overview

Nextdoor

- Built using Apache Flume
- A distributed self-hosted service for collecting and aggregating log data (similar to AWS Kinesis Streams, Apache Kafka, and Fluentd)
- Served us well for 4 years
- Scaled to handle ~1.5B events per day

# Legacy pipeline overview

Nextdoor



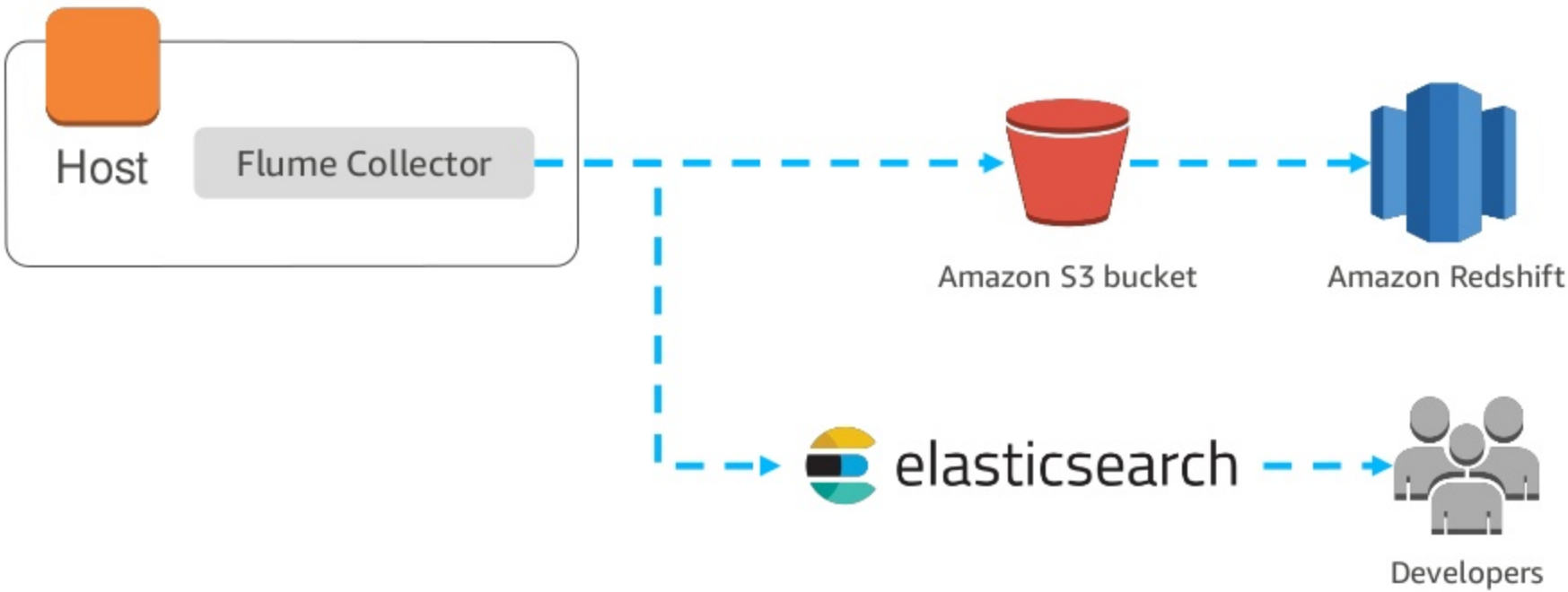




aws

# Legacy pipeline overview

Nextdoor







aws

# Legacy pipeline pain points

Nextdoor

- Highly susceptible to back pressure
  - Slow destination
  - Influx of data
  - Occasional slow Flume Collector (process, IO, CPU, bugs?)
  - Restarts were very slow (tens of minutes)
- Hard to apply changes to the pipeline
- Proprietary storage format made it difficult to near-impossible to recover corrupted data
- Flume agents were CPU hungry and this caused issues on critical infrastructure
- Hard to scale quickly with influxes of log data

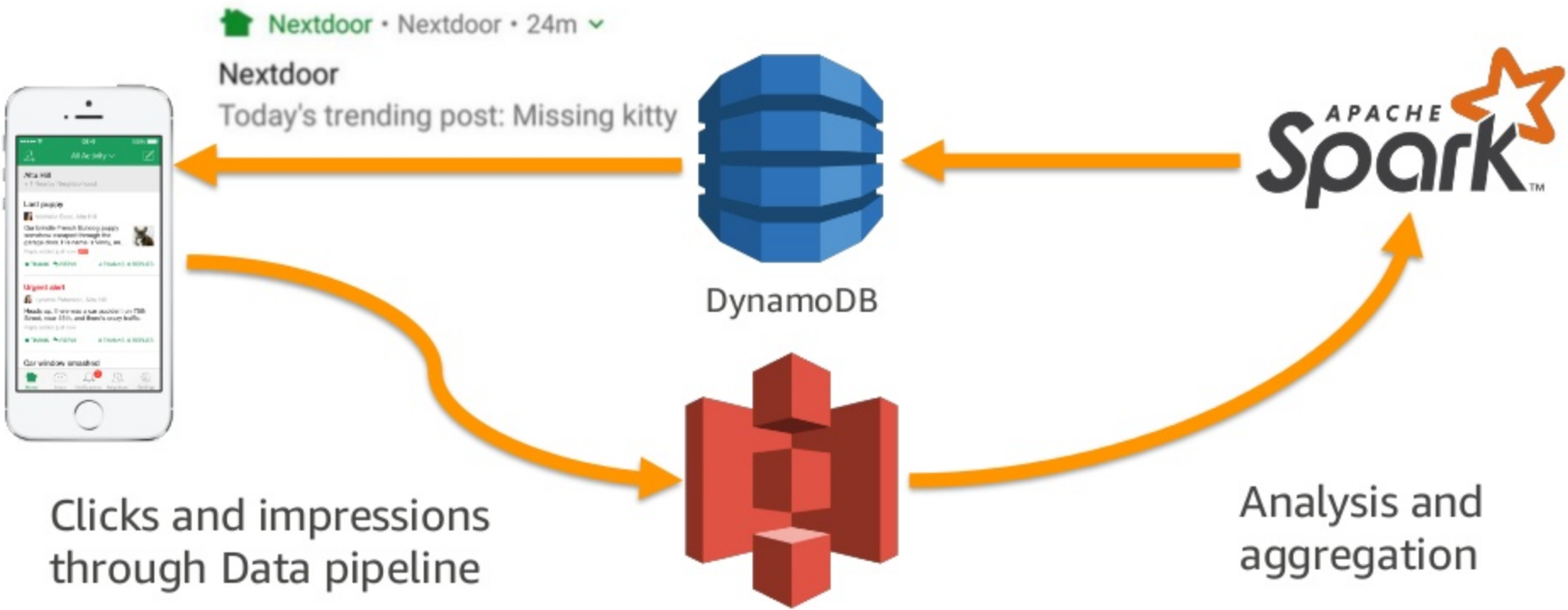
# What we wanted in a new pipeline

Nextdoor

- Spend less (or no) time managing the pipeline so we could focus on other responsibilities
- Leaner server infrastructure
- Easy (re)configuration of the pipeline
- Tighter SLAs
  - Streaming logs into Elasticsearch with an SLA of 1 minute
  - Writing partitioned batches of data into S3 with an SLA of 5 minutes
- Automatically handle influx in data and normal cyclical patterns

# Example: Trending post notifications

Nextdoor



# AWS managed offerings

Nextdoor

Kinesis Streams

Lambda

Kinesis Firehose

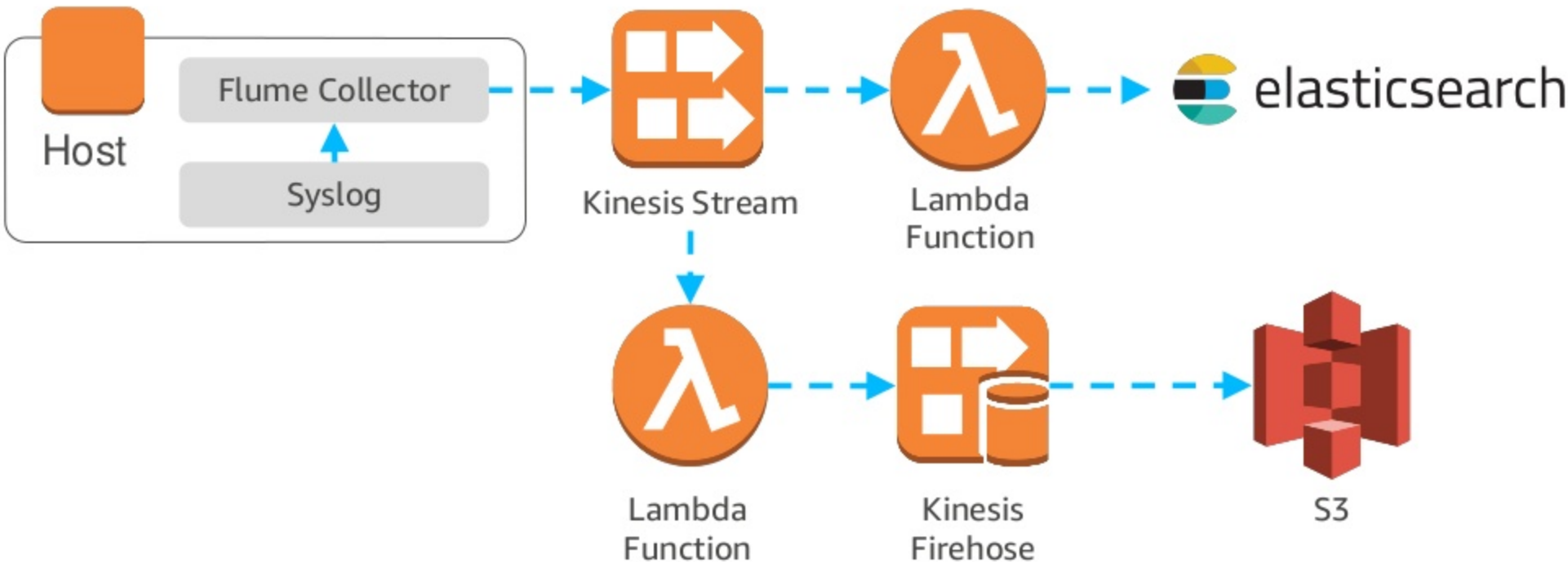
S3




aws

# What your pipeline might look like

Nextdoor







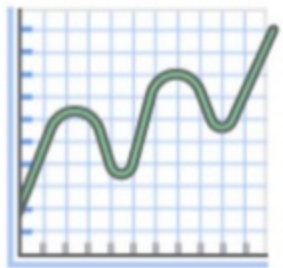
aws

# AWS Lambda functions

Nextdoor

- Easy to write lightweight processing functions in Node, Python, Java, and C#
- However, it's ETL and there's a lot of boilerplate

No Servers to Manage



Continuous Scaling


Subsecond Metering




aws

# What is ETL?

Nextdoor

**E**xtract

- Read data from a source and make available for consumption
- Consume the data into a data structure readable by a machine language (for example Java Objects)

**T**ransform

- Lowercase, Redact, Augment, Join, and so on

**L**oad

- Write object back to binary form (maybe compress)
- Transport the binary data to a destination

# Where is the boilerplate?

Nextdoor

| | Step | Example |
|---|---|---|
| Extract | Reading data source | Reading file from S3 in batches |
| | Deserialization | Binary → Object |
| Transform | Filtering | Remove trace level logging |
| | Transformation | Lowercase, redact, augment, join |
| | Serialization | Object → Binary |
| Load | Transport | Binary → Destination format → Destination |
| | Insight | Counting, logging, monitoring, and alerting |
| | Error handling | Retrying, exception handling, error reporting |

# Standardizing Lambda functions

Nextdoor

- What if tomorrow we wanted to ingest logs from ALB, ELB, VPC…
- How do we reuse existing code while adding features and minimizing spaghetti code and copypasta?

# Design considerations

Nextdoor

- CPU cycles
- Memory footprint
- Network I/O
- Concurrency and thread pooling
- Buffering and compressing
- Retry logic
- Graceful error handling
- Metric reporting
- Configuration

# Our solution

Nextdoor


**github.com/nextdoor/bender**

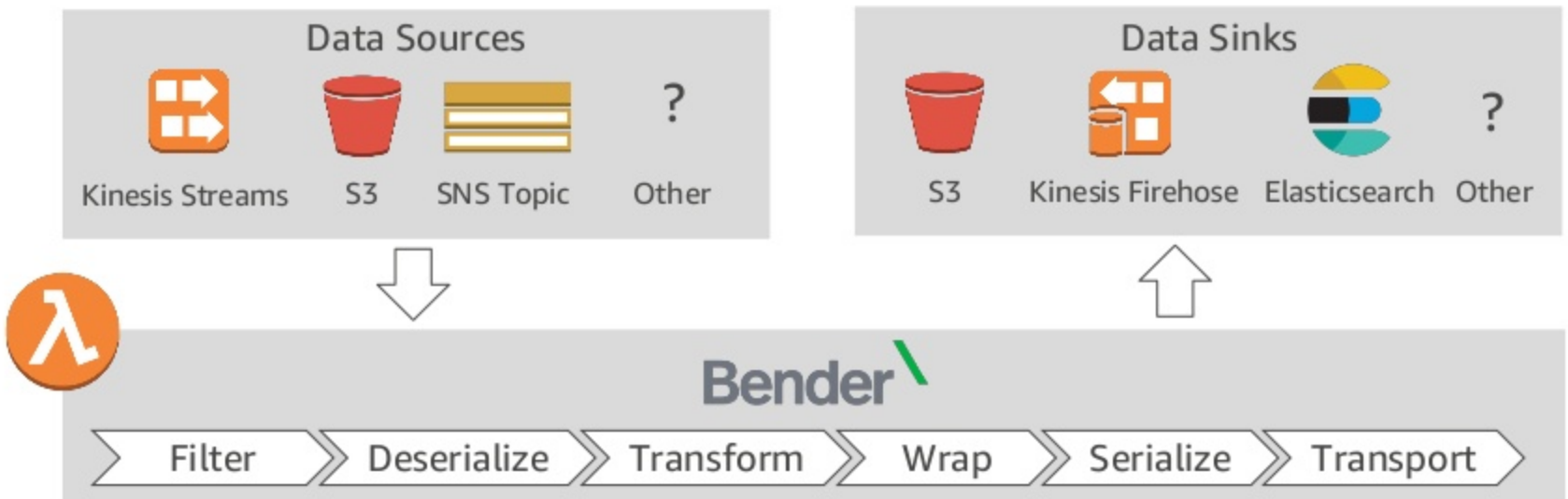



aws

# No more spaghetti and copypasta!

Nextdoor

Bender handles the complex plumbing and provides the interfaces necessary to build your own business logic for any step of the ETL process.



# What ETL steps are supported?

Nextdoor

| | ETL Step | |
|---|---|---|
| Extract | Sources | Kinesis, S3, and SNS |
| | Deserialization | JSON and anything parseable by a RegEx |
| Transform | Filtering | RegEx and str contains |
| | Operations | GeoIp lookup, multi-record splits, Elasticsearch Ingest |
| | Wrappers | Basic, S3, and Kinesis |
| Load | Transporters | Firehose, S3, Elasticsearch, Splunk, Sumologic, Scalyr, Generic HTTP |

# Additional functionality

Nextdoor

| | Other | |
|---|---|---|
| Useful | Error handling | Graceful retries and robust exception handling |
| | Reporting | CloudWatch metrics and Datadog |
| | Configuration | Config validation before deployment |
| | Documentation | Auto generated documentation from code |

# Bender: Configuration

Nextdoor

# What does the configuration look like?

Nextdoor

```
handler:
  type: KinesisHandler
sources:
- name: Events
  source_regex: .*:stream/my-stream
  regex_patterns: [".*TRACE.*"]
  deserializer:
     type: GenericJson
  operations:
  - ...
wrapper:
  type: KinesisWrapper
serializer:
  type: Json
transport:
  type: ElasticSearch
  index_time_format: yyyy-MM-dd
reporters:
- type: Cloudwatch
```

# What does the configuration look like?

Nextdoor

```
handler:
  type: KinesisHandler
sources:
- name: Events
  source_regex: .*:stream/my-stream
  regex_patterns: [".*TRACE.*"]
  deserializer:
     type: GenericJson
  operations:
  - ...
wrapper:
  type: KinesisWrapper
serializer:
  type: Json
transport:
  type: ElasticSearch
  index_time_format: yyyy-MM-dd
reporters:
- type: Cloudwatch
```

Type of Lambda trigger this function is using

Kinesis: prod-syslog

arn:aws:kinesis:us-east-1:123456789:stream/prod-syslog

Last processing result: **OK** Batch size: **10000**

# What does the configuration look like?

Nextdoor

```
handler:
  type: KinesisHandler
sources:
- name: Events
  source_regex: .*:stream/my-stream
  regex_patterns: [".*TRACE.*"]
  deserializer:
     type: GenericJson
  operations:
  - ...
wrapper:
  type: KinesisWrapper
serializer:
  type: Json
transport:
  type: ElasticSearch
  index_time_format: yyyy-MM-dd
reporters:
- type: Cloudwatch
```

How to treat data coming from the source

# What does the configuration look like?

Nextdoor

```
handler:
  type: KinesisHandler
sources:
- name: Events
  source_regex: .*:stream/my-stream
  regex_patterns: [".*TRACE.*"]
  deserializer:
     type: GenericJson
  operations:
  - ...
wrapper:
  type: KinesisWrapper
serializer:
  type: Json
transport:
  type: ElasticSearch
  index_time_format: yyyy-MM-dd
reporters:
- type: Cloudwatch
```





aws

# What does the configuration look like?

Nextdoor

```
handler:
  type: KinesisHandler
sources:
- name: Events
  source_regex: .*:stream/my-stream
  regex_patterns: [".*TRACE.*"]
  deserializer:
    type: GenericJson
  operations:
  - ...
wrapper:
  type: KinesisWrapper
serializer:
  type: Json
transport:
  type: ElasticSearch
  index_time_format: yyyy-MM-dd
reporters:
- type: Cloudwatch
```

How to read data coming from the source

# What does the configuration look like?

Nextdoor

```
handler:
  type: KinesisHandler
sources:
- name: Events
  source_regex: .*:stream/my-stream
  regex_patterns: [".*TRACE.*"]
  deserializer:
     type: GenericJson
  operations:
  - ...
wrapper:
  type: KinesisWrapper
serializer:
  type: Json
transport:
  type: ElasticSearch
  index_time_format: yyyy-MM-dd
reporters:
- type: Cloudwatch
```

Operations to perform on the data

# What does the configuration look like?

Nextdoor

```
handler:
  type: KinesisHandler
sources:
- name: Events
  source_regex: .*:stream/my-stream
  regex_patterns: [".*TRACE.*"]
  deserializer:
     type: GenericJson
  operations:
  - ...
wrapper:
  type: KinesisWrapper
serializer:
  type: Json
transport:
  type: ElasticSearch
  index_time_format: yyyy-MM-dd
reporters:
- type: Cloudwatch
```

Add a wrapper

```
{
    "partition_key": "1234.456",
    "sequence_number": "12345",
    "source_arn": "us-east-1:1234:stream/logs.",
    "event_source": "aws:kinesis",
    "function_name": "log-pipeline...",
    "function_version": "97",
    "processing_time": 1507823274129,
    "arrival_time": 1507823273628,
    "processing_delay": 1199,
    "timestamp": 1507823272930,
    "payload": {
       // JSON data from original message
    }
}
```




aws

# What does the configuration look like?

Nextdoor

```
handler:
  type: KinesisHandler
sources:
- name: Events
  source_regex: .*:stream/my-stream
  regex_patterns: [".*TRACE.*"]
  deserializer:
     type: GenericJson
  operations:
  - ...
wrapper:
  type: KinesisWrapper
serializer:
  type: Json
transport:
  type: ElasticSearch
  index_time_format: yyyy-MM-dd
reporters:
- type: Cloudwatch
```

Write it back to JSON string or maybe something else?

# What does the configuration look like?

Nextdoor

```
handler:
  type: KinesisHandler
sources:
- name: Events
  source_regex: .*:stream/my-stream
  regex_patterns: [".*TRACE.*"]
  deserializer:
     type: GenericJson
  operations:
  - ...
wrapper:
  type: KinesisWrapper
serializer:
  type: Json
transport:
  type: ElasticSearch
  index_time_format: yyyy-MM-dd
reporters:
- type: Cloudwatch
```

Send it off somewhere




aws

# What does the configuration look like?

Nextdoor

```
handler:
  type: KinesisHandler
sources:
- name: Events
  source_regex: .*:stream/my-stream
  regex_patterns: [".*TRACE.*"]
  deserializer:
     type: GenericJson
  operations:
  - ...
wrapper:
  type: KinesisWrapper
serializer:
  type: Json
transport:
  type: ElasticSearch
  index_time_format: yyyy-MM-dd
reporters:
- type: Cloudwatch
```

Report metrics

# Extending Bender

Nextdoor

# Operation to transform JSON

Nextdoor

If key exists replace with value

```
if payload.has_key(key):
   payload[key] = new_value
```





aws

# What the config code will look like

Nextdoor

```
@JsonTypeName("JsonReplaceKVOperation")
@JsonSchemaDescription("Replaces a given key with specified value.")
public class JsonReplaceKVOperationConfig extends OperationConfig {

    @JsonSchemaDescription("Key name to find")
    @JsonProperty(required = true)
    private String key;

    @JsonSchemaDescription("New string value")
    @JsonProperty(required = true)
    private String value;

    @Override
    public Class<JsonReplaceKVOperationFactory> getFactoryClass() {
    return JsonReplaceKVOperationFactory.class;
    }

    // getters and setters
}
```

# What the config file will look like

Nextdoor

```
handler:
  type: KinesisHandler
sources:
- name: Events
  source_regex: .*:stream/my-stream
  deserializer:
     type: GenericJson
  operations:
  - type: JsonReplaceKVOperation
    key: foo
    value: bar
...
```

# What the documentation will look like

Nextdoor

JsonReplaceKVOperation

Replaces a given key with specified value.

***type*** req

string constant JsonReplaceKVOperation default JsonReplaceKVOperation

***key*** req string

Key name to find

***value*** req string

New string value

# Operation code

Nextdoor

```
public class JsonReplaceKVOperation extends PayloadOperation {
  private String key;
  private String value;

  public JsonReplaceKVOperation(String key, String value) {
     this.key = key;
     this.value = value;
  }

  @Override
  protected void perform(JsonObject payload) {
     if (payload.has(this.key))
        // Note: addProperty replaces the current value
        payload.addProperty(this.key, this.value);
  }
}
```

# Code on GitHub

Nextdoor

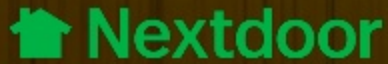

**github.com/nextdoor/bender**




aws

# Why did we choose AWS serverless offerings?

Matt Wise · Sr. Systems Architect



# Kinesis for data ingestion

Nextdoor

- Operational overhead is near zero
- Scaling streams up and down is a simple API call
- Scaling up gives you more performance in every dimensions
- Data retention of 24 hours to 7 days
- Stream access is fast – delays measured in milliseconds
- AWS provides tools…
    - Log collection agent
    - Stream Auto Scaling agent
    - KCL for in-app direct stream publishing

# Lambda for our execution layer

Nextdoor

- Scales automatically with Kinesis and your data
- Manages Kinesis Triggers, Function Iterators all on its own
- Provides plenty of metrics that are easy to alarm on
- Blazing fast … backlogs are worked through as fast as your target can handle it
- Built in easy to use management operations
    - Pausing and enabling triggers
    - Releasing by updating a function to the latest version
    - Rollbacks by just rolling back to the last known good version

# Firehose for data aggregation

Nextdoor

- Lambda+Kinesis Streams = Lots of small operations!
- Kinesis -> Lambda -> S3 = Tons of tiny files
- Bulk-uploading files to S3 leads to faster downstream analytics performance
- Bender can batch multiple individual events up into a single Firehose Record, keeping the costs sane and maximizing performance!
- Max-file-size and Max-event-age settings allow you to control your uploads and your data delay

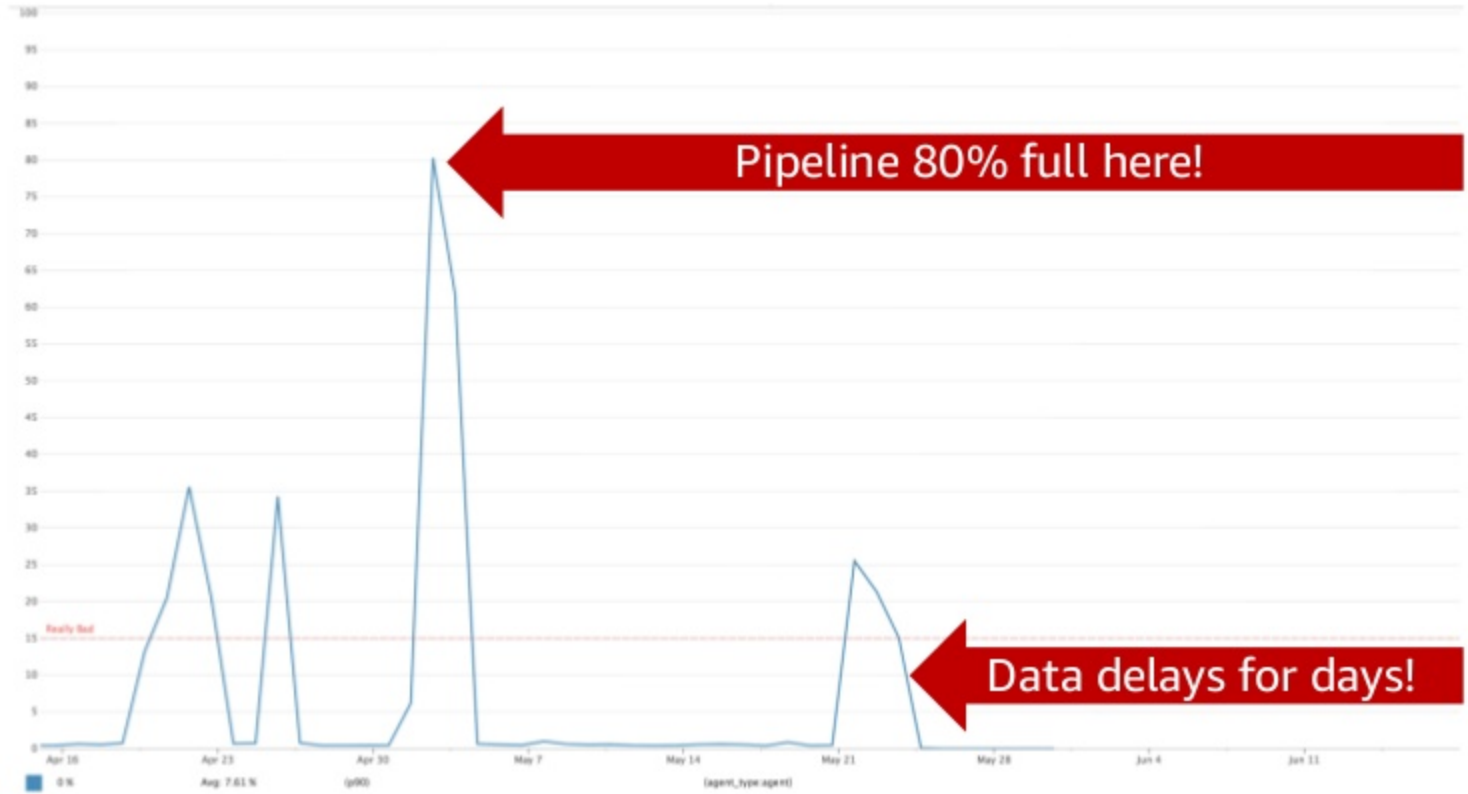
# S3 for short term and long term storage

Nextdoor

- Fast, cost effective, and stable
- Lifecycle policies allow for automatic purging or migration of data
- Hands-off operation
- Fine grained access controls
- And so on, and so on, and so on...

# Final months of the old pipeline…

Nextdoor



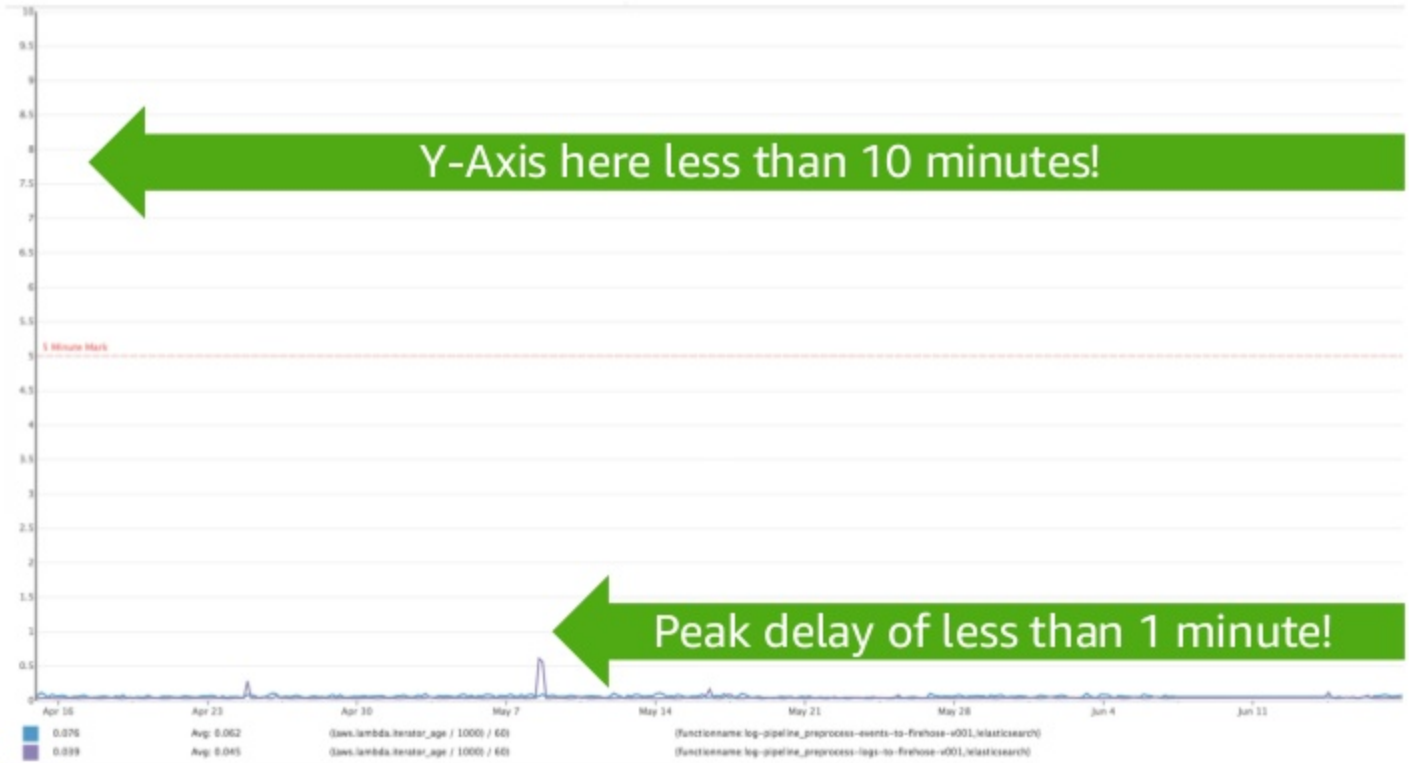




aws

# Well that looks better!

Nextdoor



# Totally worth it!

Nextdoor

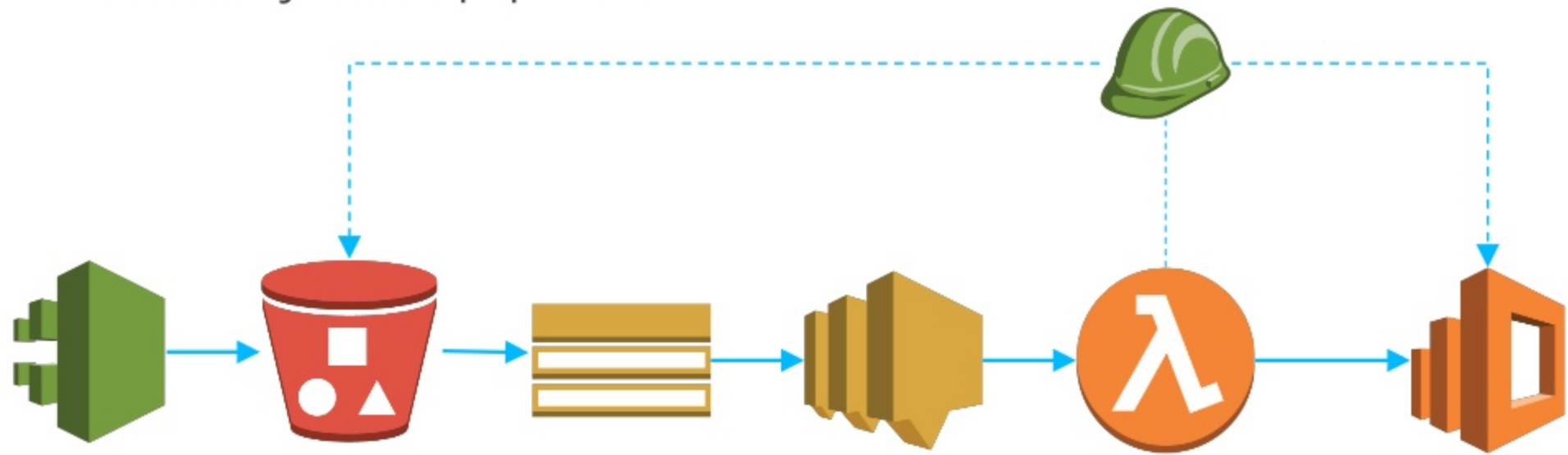
| Flume | Kinesis/Lambda/Firehose/S3 |
|---|---|
| @ 1.5B events/day, was dropping 0.02% | @ 3B events/day, all events make it |
| Required significant architectural work to scale beyond ~2B events/day | Built to scale on day one, all major scaling components are AWS-managed |
| Near weekly outages that lasted for hours | In 12 months, 1 outage that lasted more than an hour.. Automatically resolved by AWS |
| Significant operational knowledge required for operation and scaling of the system | Java knowledge required to add features – AWS operates the rest |
| ~15-20% of a an engineers time to manage | Literally almost zero management in months |
| Combined ingest/storage/forwarding make changes hard and reduce stability | Separated responsibilities are easier to scale and tweak, improve reliability |

A real pipeline!

Nextdoor

# Anatomy of a pipeline

Nextdoor



CloudTrail S3 Bucket SNS Topic Subscription Bender ElasticSearch Domain

Trails > Configuration

Logging ON

cloudtrail-bender-demo-Cloudtrail-1U3VDHRF3JUJL

## Trail settings

When a trail applies to all regions, the trail exists in all regions and delivers log files for all regions to one Amazon S3 bucket and an optional CloudWatch Logs log group. To see all of your trails, click Trails.

**Apply trail to all regions** Yes

## Management events

## Data events

## Storage location

**S3 bucket** cloudtrail-bender-demo

**Last log file delivered** 2017-10-20, 3:17 pm

**Encrypt log files** No

**Enable log file validation** No

**Publish to SNS** No

Amazon S3 > cloudtrail-bender-demo / AWSLogs / 364942603424 / CloudTrail / us-east-1 / 2017 / 10 / 20

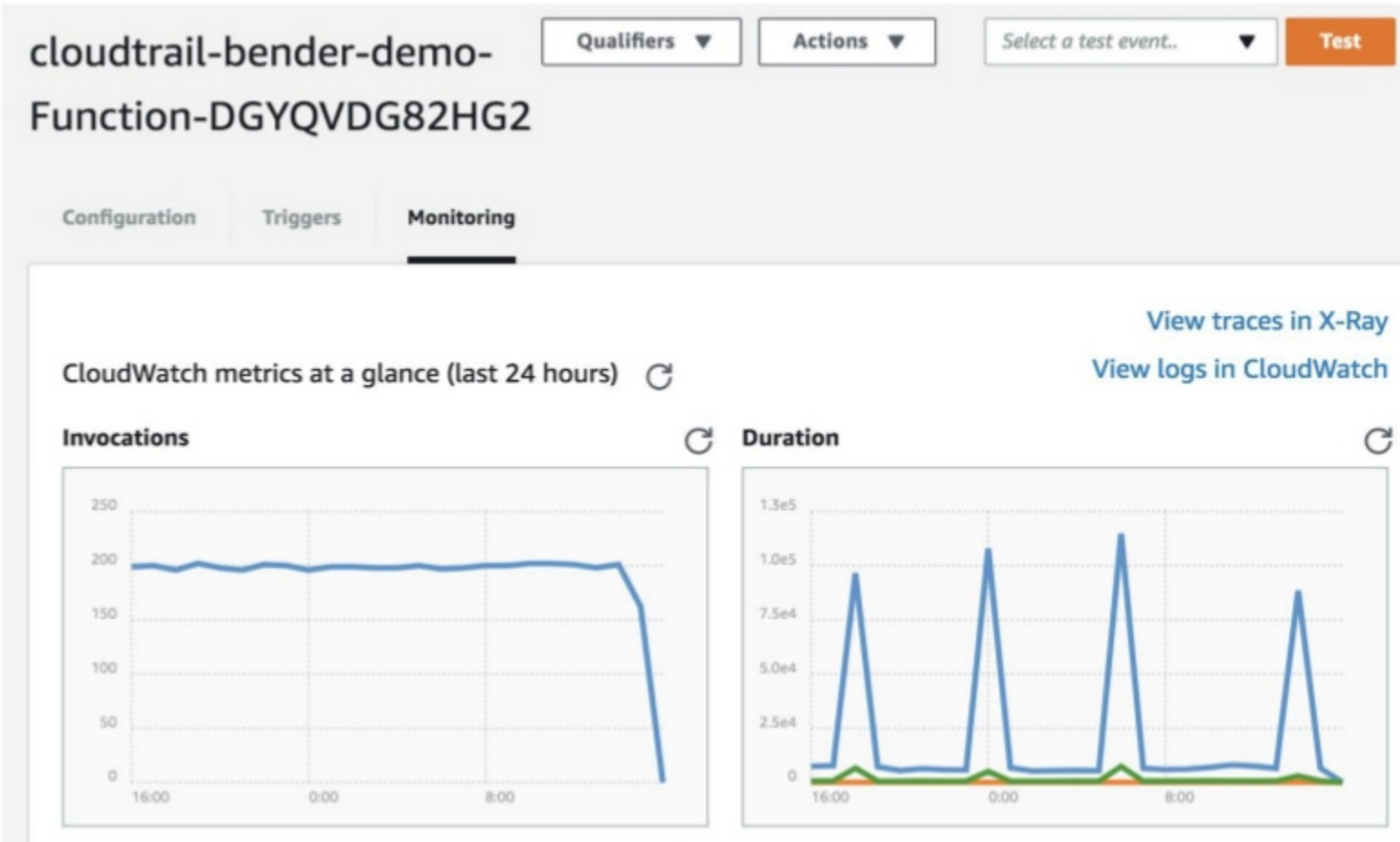
Overview

Type a prefix and press Enter to search. Press ESC to clear.

Upload + Create folder More US East (N. Virginia)

Viewing 1 to 300 >

| Name | Last modified | Size | Storage class |
|---|---|---|---|
| 364942603424_CloudTrail_us-east-1_20171020T0000Z_UlR0fJ... | Oct 19, 2017 5:02:32 PM | 60.1 KB | Standard |
| 364942603424_CloudTrail_us-east-1_20171020T0000Z_uyotu4... | Oct 19, 2017 5:02:27 PM | 38.3 KB | Standard |
| 364942603424_CloudTrail_us-east-1_20171020T0005Z_7tqInn... | Oct 19, 2017 5:07:31 PM | 26.5 KB | Standard |
| 364942603424_CloudTrail_us-east-1_20171020T0005Z_CaZ7Q... | Oct 19, 2017 5:07:40 PM | 39.7 KB | Standard |
| 364942603424_CloudTrail_us-east-1_20171020T0010Z_BvhYfJ... | Oct 19, 2017 5:12:34 PM | 21.9 KB | Standard |
| 364942603424_CloudTrail_us-east-1_20171020T0010Z_frlhQ3t... | Oct 19, 2017 5:12:38 PM | 17.0 KB | Standard |
| 364942603424_CloudTrail_us-east-1_20171020T0015Z_LetfqJ... | Oct 19, 2017 5:17:38 PM | 34.8 KB | Standard |



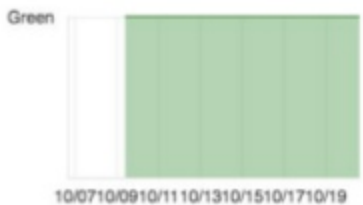
# cloudtrail-bender-demo

Configure cluster | Modify access policy | Manage tags | Delete domain

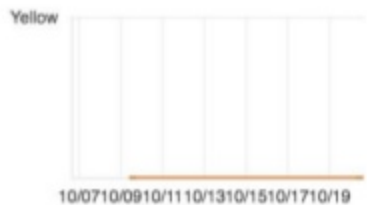
Overview | Cluster health | Indices | Monitoring | Logs

Time range: Last 2 weeks

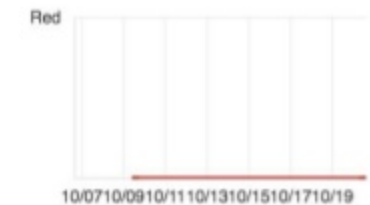
**ClusterStatus.green** (Count)
Statistic: Minimum



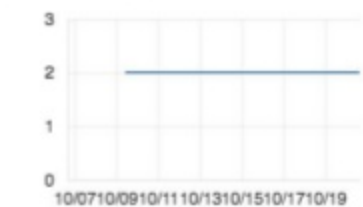

**ClusterStatus.yellow** (Count)
Statistic: Maximum



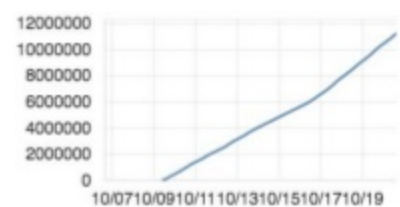

**ClusterStatus.red** (Count)
Statistic: Maximum



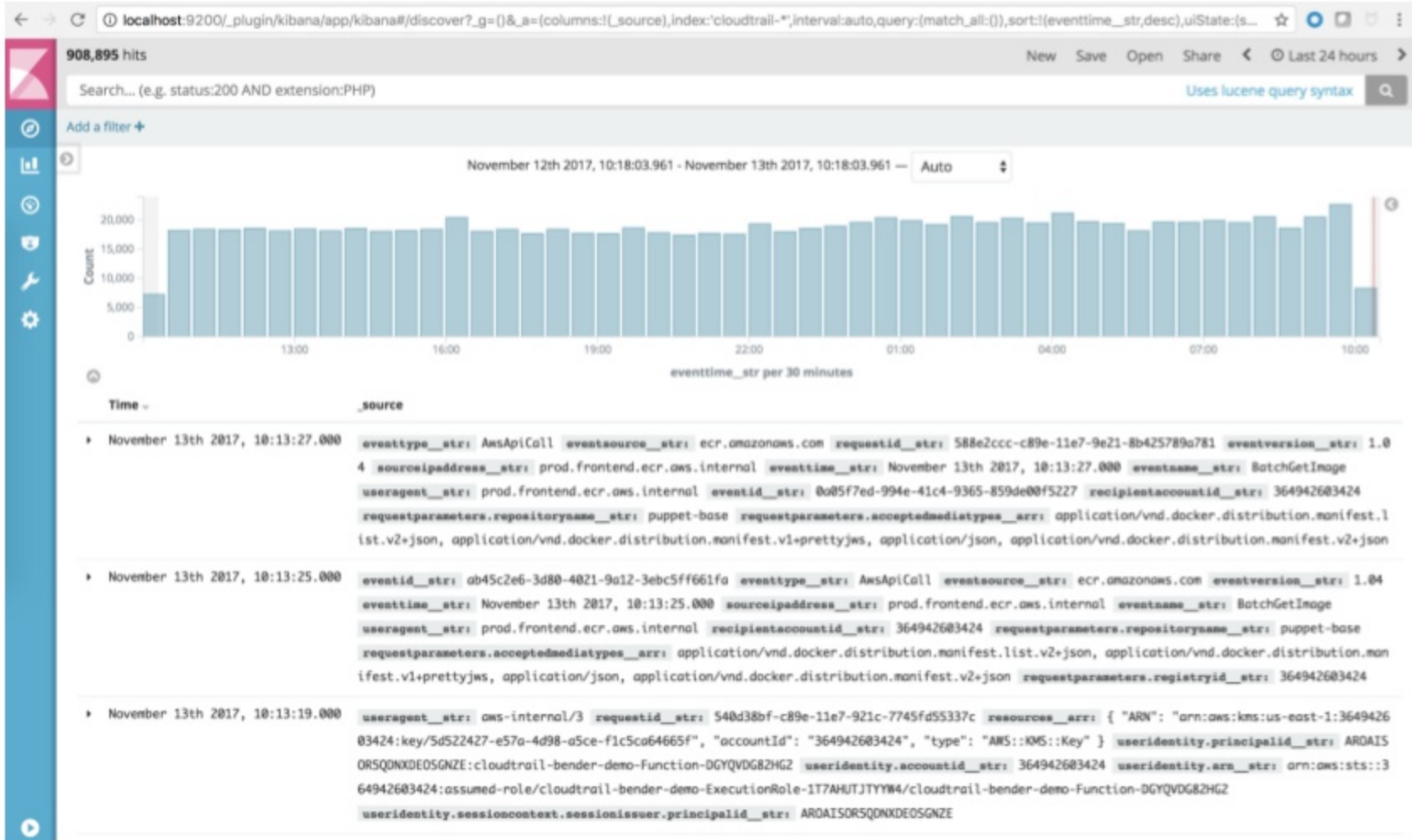

**Nodes** (Count)
Statistic: Average




**SearchableDocuments** (Count)
Statistic: Average




**DeletedDocuments** (Count)
Statistic: Average



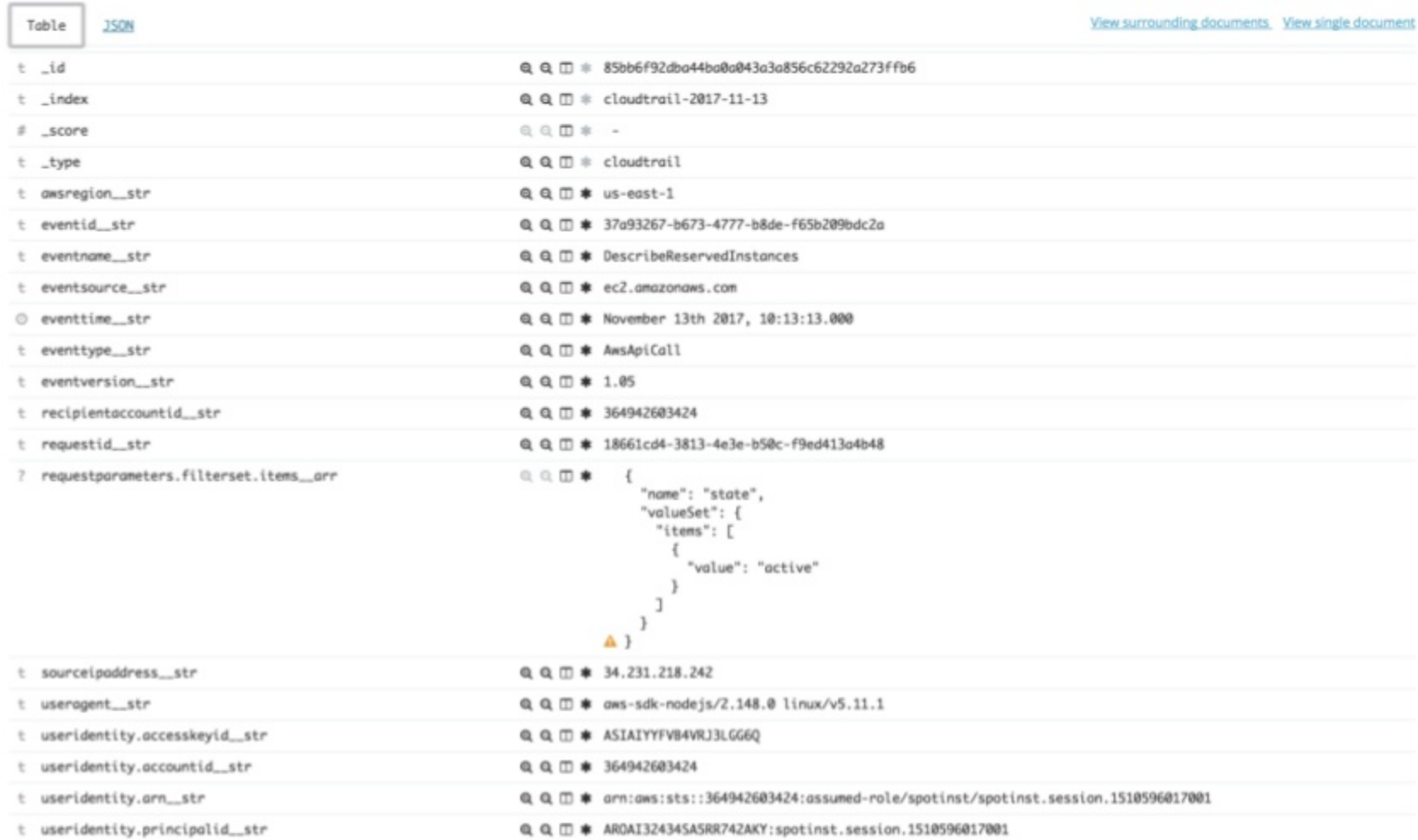
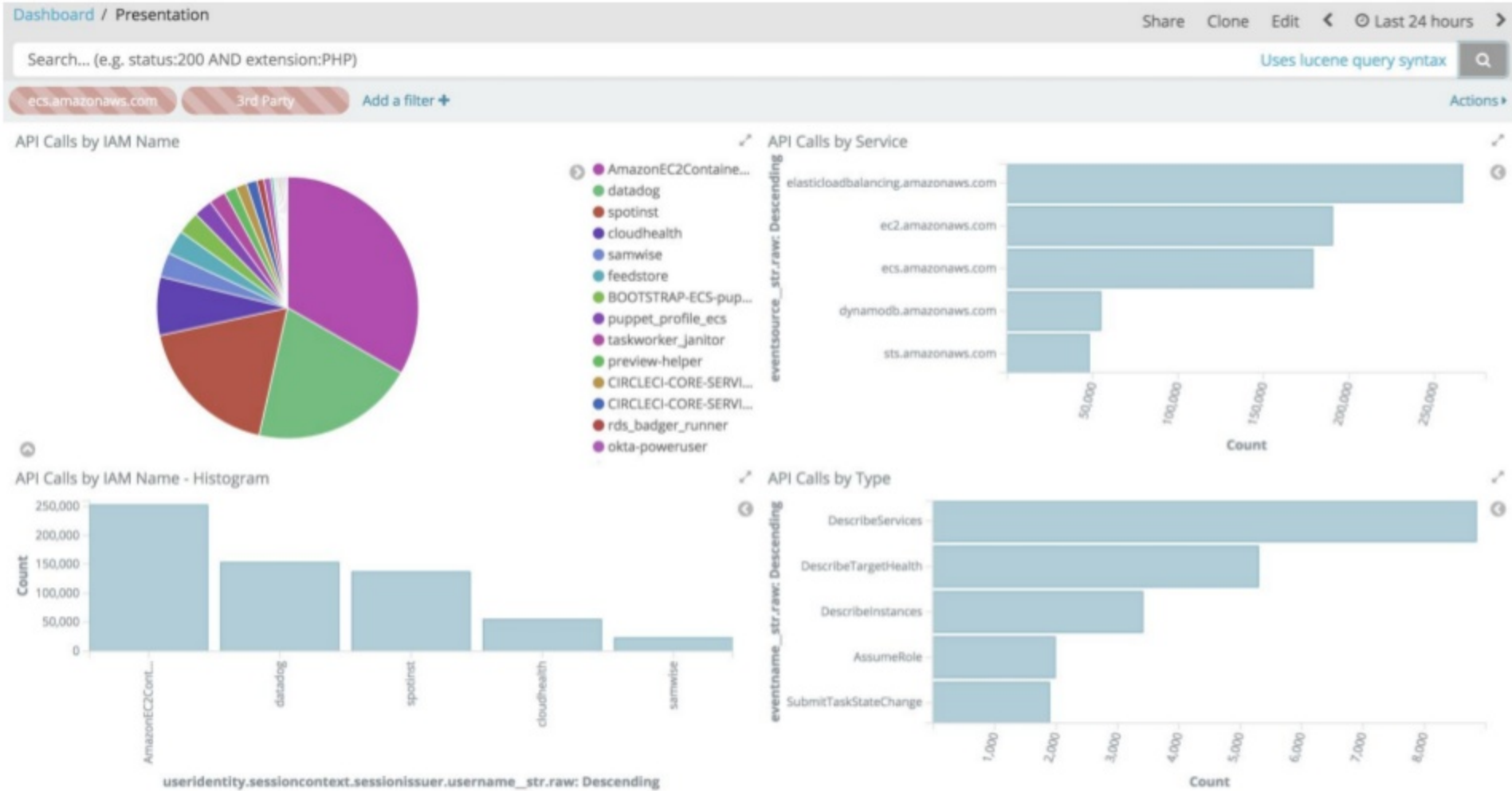


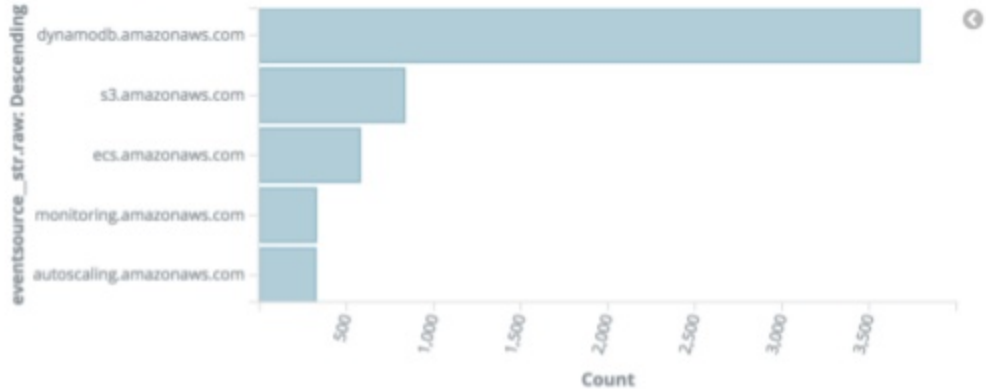
Table JSON View surrounding documents View single document

| | | |
|---|---|---|
| t | _id | 85bb6f92dba44ba0a043a3a856c62292a273ffb6 |
| t | _index | cloudtrail-2017-11-13 |
| # | _score | - |
| t | _type | cloudtrail |
| t | awsregion__str | us-east-1 |
| t | eventid__str | 37a93267-b673-4777-b8de-f65b209bdc2a |
| t | eventname__str | DescribeReservedInstances |
| t | eventsource__str | ec2.amazonaws.com |
| t | eventtime__str | November 13th 2017, 10:13:13.000 |
| t | eventtype__str | AwsApiCall |
| t | eventversion__str | 1.05 |
| t | recipientaccountid__str | 364942603424 |
| t | requestid__str | 18661cd4-3813-4e3e-b50c-f9ed413a4b48 |
| ? | requestparameters.filterset.items__arr | { "name": "state", "valueSet": { "items": [ { "value": "active" } ] } } |
| t | sourceipaddress__str | 34.231.218.242 |
| t | useragent__str | aws-sdk-nodejs/2.148.0 linux/v5.11.1 |
| t | useridentity.accesskeyid__str | ASIAIYYFVB4VRJ3LGG6Q |
| t | useridentity.accountid__str | 364942603424 |
| t | useridentity.arn__str | arn:aws:sts::364942603424:assumed-role/spotinst/spotinst.session.1510596017001 |
| t | useridentity.principalid__str | AROAI32434SA5RR742AKY:spotinst.session.1510596017001 |



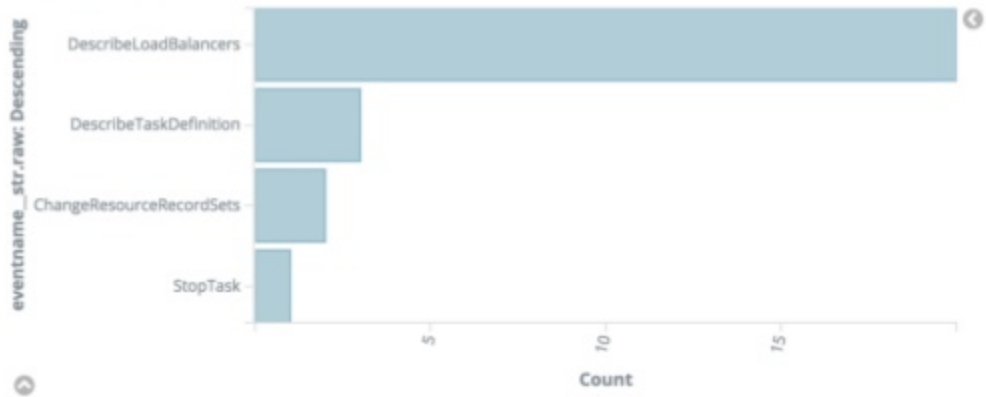
API Calls by IAM Name

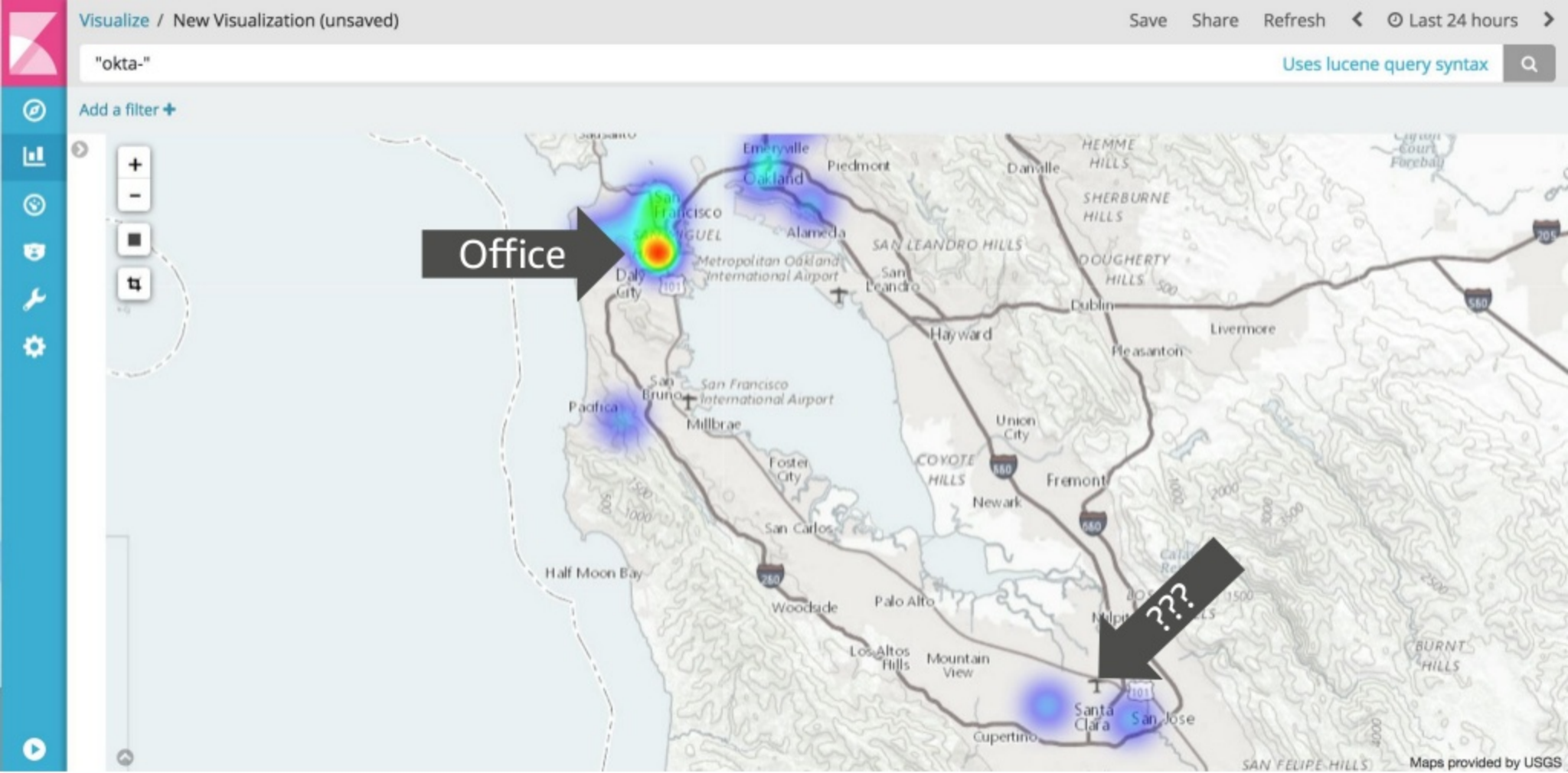

API Calls by Service

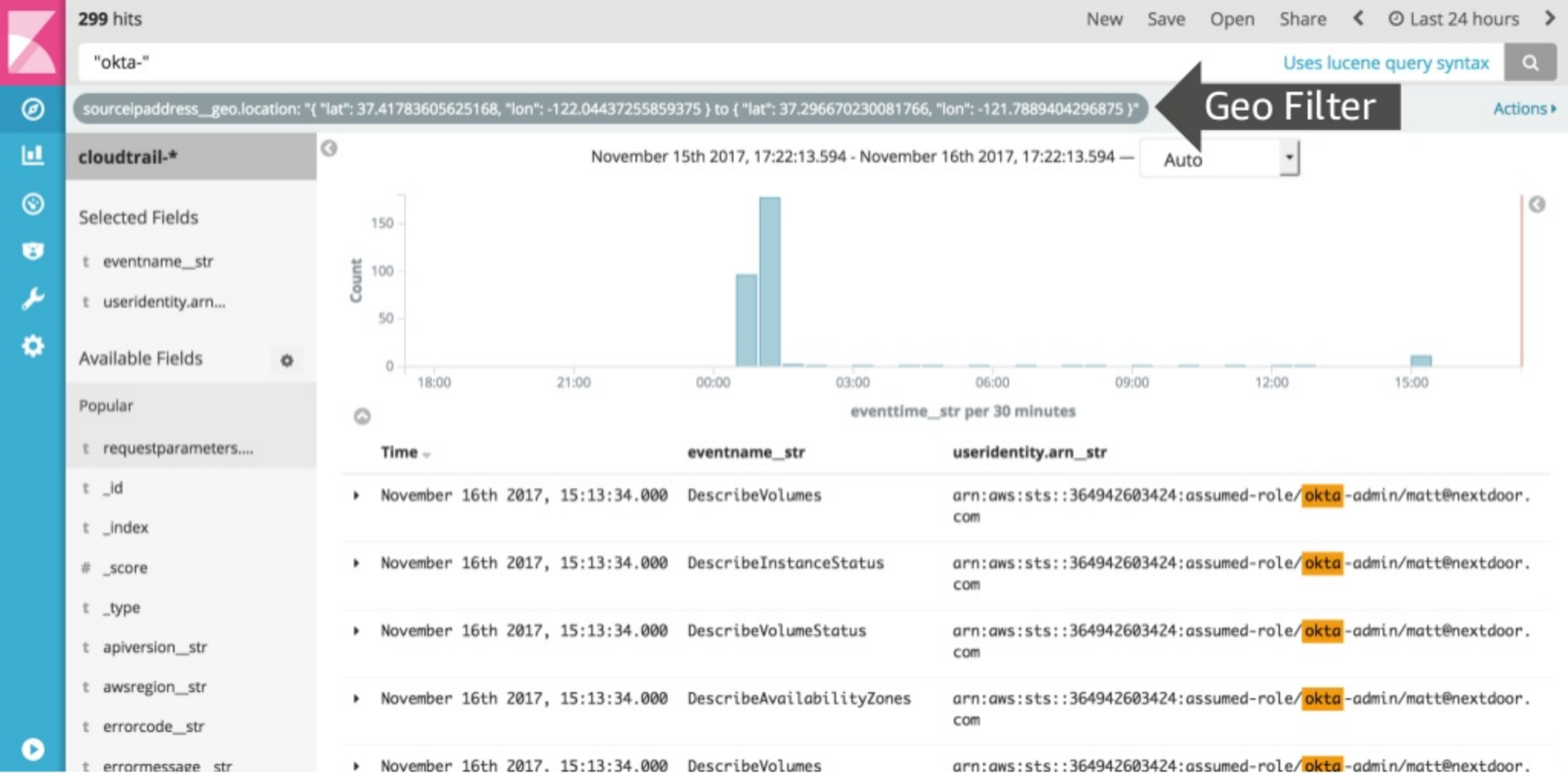


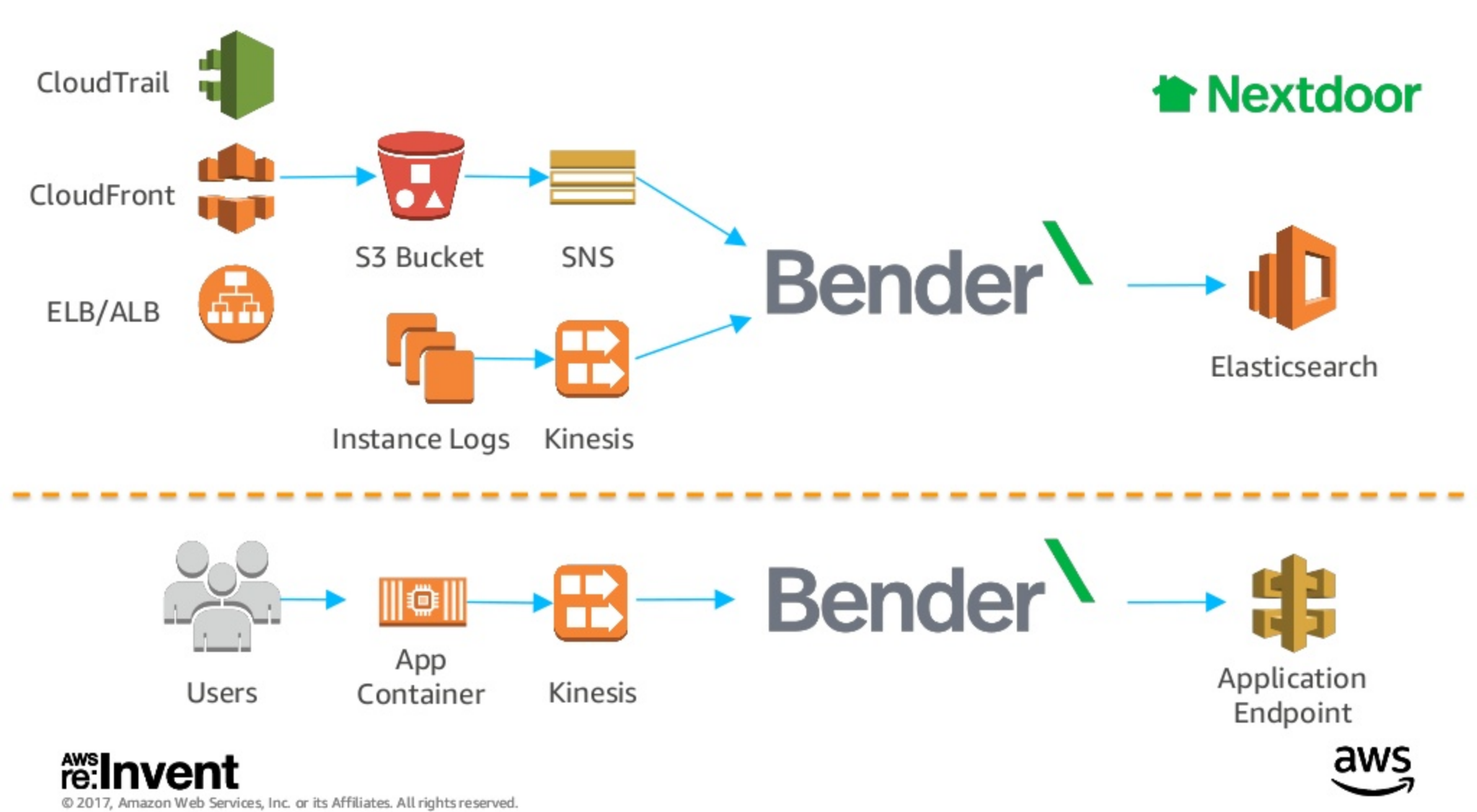

API Calls by IAM Name - Histogram


API Calls by Type









# Final thoughts…

Nextdoor

- The Systems Team @ Nextdoor has 3 people…
- We manage hundreds of servers, and we *automate everything*
- Hands-off auto-scaling is critical to keeping our team small and efficient
- Lambda/Kinesis costs are a drop in the bucket compared to engineering productivity costs when things don't work

# We're hiring!

Nextdoor

**nextdoor.com/jobs**

https://s3.amazonaws.com/bender-presentation-assets-reinvent-2017/bender-stack.yaml


AWS re:Invent



aws