StreamAlert

Serverless, Real-Time Data Analysis

airbnb

@jack_naglieri / Enigma 2017

Hypothetical: You just joined a new team, and need to collect, analyze, and alert on log data.

- Two colleagues on your team
- Thousands of laptops + production servers
- Must keep up with growth

Option 1: Develop and deploy your own tool

Option 1 - Develop and deploy your own tool

Challenges

- Engineering time and resources
- Responsible for:
  - Reliability
  - Security
  - Scalability

Have you had to rebuild a tool that you previously created?

Option 2: Deploy an existing tool - open source or commercial

Option 2 - Deploy an existing tool

Challenges

- Customizations necessary
- Scaling and upgrading are non-trivial
- Deployment challenges:
  - Time
  - Skillset required
  - Reliance on other teams

Has cost, time, or staffing prevented you from deploying a tool you needed?

Ideal Option

- Automated deployment
- Low operational overhead
- Built-in scalability and reliability
- Secure by default

Getting There

Cloud Infrastructure

Infrastructure as code

streamalert

What is StreamAlert?

- Serverless, real-time data analysis
- Point-in-time alerting
- Customizable to meet your needs

Benefits of StreamAlert

- Scalable to TBs/day
- Automated deployment
- Minimal system ownership
- Rules written in Python
- Low cost

What type of data can StreamAlert analyze?

JSON

{"name":"logged_in_users", "host":"ubuntu", "calendarTime":"Jan 10 17:49:07","columns":{"host":"10.0.0.2","username":"vagrant"}}

Syslog

Jan 10 17:49:07 ubuntu sshd[9644]: Accepted publickey for vagrant from 10.0.2.2 port 56738 ssh2

What type of data can StreamAlert analyze?

CSV

2,123456789010,eth0,10.0.0.1,10.0.0.2,56738,22,6,20,4249,ACCEPT,OK

Key Value

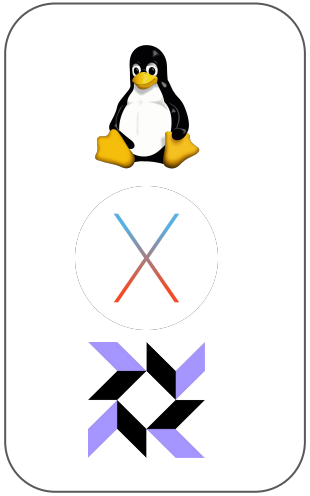
msg=audit(1364475353.159:24270): user pid=3280 uid=100 auid=500 ses=1

msg='op=PAM:authentication res=success

Example Logs

Environment

System



Network

[Web] Application

streamalert

Make the deployment of security tools simple.

Design

Data Analysis

Rules

Alerts

Deployment

Compute
- EC2
- EC2 Container Service
- Lightsail
- Elastic Beanstalk
- Lambda

Storage
- S3
- Elastic File System
- Glacier
- Storage Gateway

Database
- RDS
- DynamoDB
- ElastiCache
- Redshift

Networking & Content Delivery
- VPC
- CloudFront
- Direct Connect
- Route 53

Migration
- DMS
- Server Migration
- Snowball

Developer Tools
- CodeCommit
- CodeBuild
- CodeDeploy
- CodePipeline

Management Tools
- CloudWatch
- CloudFormation
- CloudTrail
- Config
- OpsWorks
- Service Catalog
- Trusted Advisor

Security, Identity & Compliance
- IAM
- Inspector
- Certificate Manager
- Directory Service
- WAF & Shield
- Compliance Reports

Analytics
- Athena
- EMR
- CloudSearch
- Elasticsearch Service
- Kinesis
- Data Pipeline
- QuickSight

Artificial Intelligence
- Lex
- Polly
- Rekognition
- Machine Learning

Internet Of Things
- AWS IoT

Game Development
- GameLift

Mobile Services
- Mobile Hub
- Cognito
- Device Farm
- Mobile Analytics
- Pinpoint

Application Services
- Step Functions
- SWF
- API Gateway
- AppStream
- Elastic Transcoder

Messaging
- SQS
- SNS
- SES

Business Productivity
- WorkDocs
- WorkMail

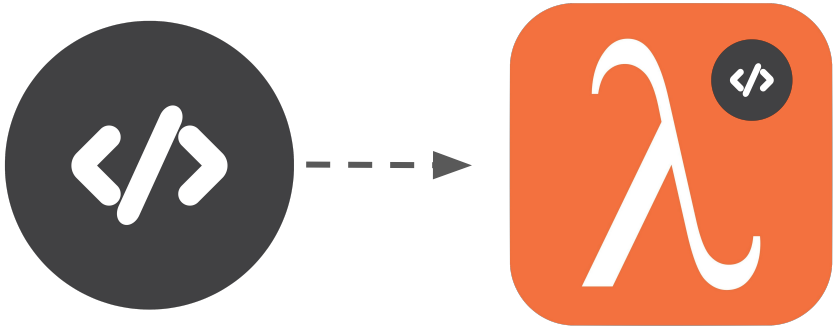
Desktop & App Streaming
- WorkSpaces
- AppStream 2.0

amazon web services

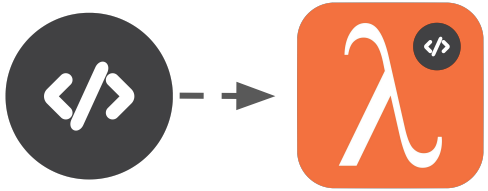
Serverless - Focus on the application logic, not the servers

Serverless Compute Model

1. Write Application
2. Upload to AWS Lambda
3. Run



Serverless Compute Pricing Model



compute + # of requests = total cost

duration: 100ms

memory: 128MB

1,000,000 req/day

$5.80/month

Built-in Security Benefits

1. Role Based Access Control via AWS IAM
2. Natural data segmentation
3. Isolated (containerized) log analysis
4. TLS

Design

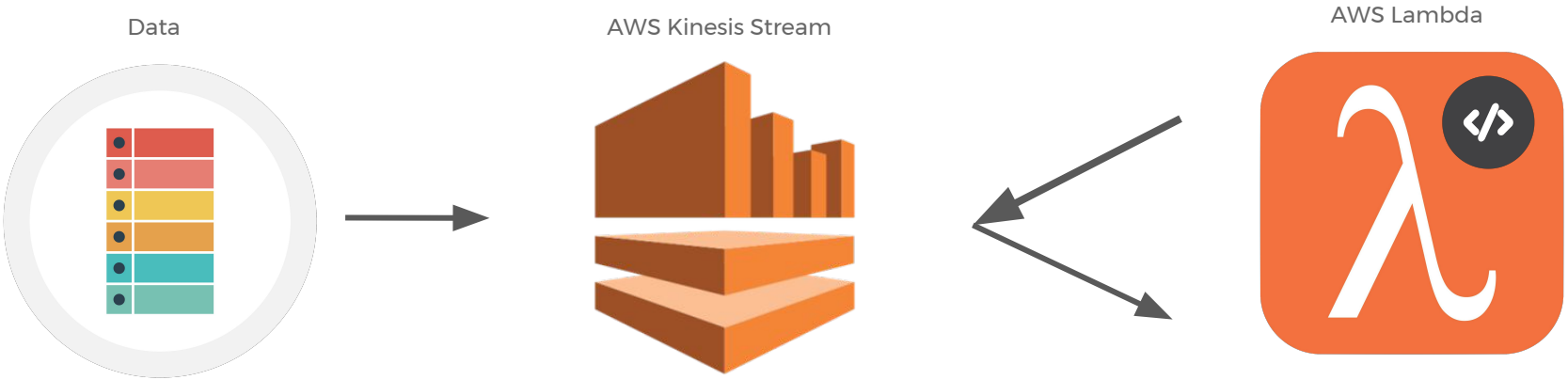
Data Analysis

Rules

Alerts

Deployment

High Level



Data is sent to a Kinesis Stream; Lambda polls the stream and analyzes the data

AirbnbEng

Creative engineers and data scientists building a world where you can belong anywhere. http://airbn...

May 3 · 5 min read

Introducing Syslog to AWS Kinesis via Osquery

Logs awaiting collection (Logs in Yyteri by kallerna, licensed under Creative Commons)

osquery queries run on hosts

```
SELECT * FROM users;
SELECT * FROM processes;
SELECT * FROM syslog ...;
SELECT * FROM process_open_sockets ...;
```

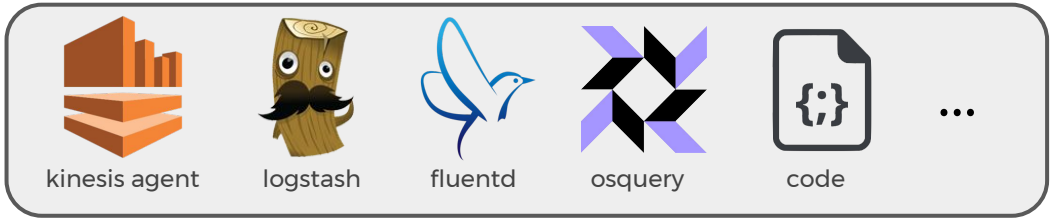
resulting data

```
{
  "hostIdentifier": "web01",
  "calendarTime": "Aug 10 10:13:54”
  "columns": {
    "remote_address": "51.32.104.190",
    "remote_port": "22",
    ...
}
...
```

AWS Kinesis Stream

AWS Lambda

Sending Data

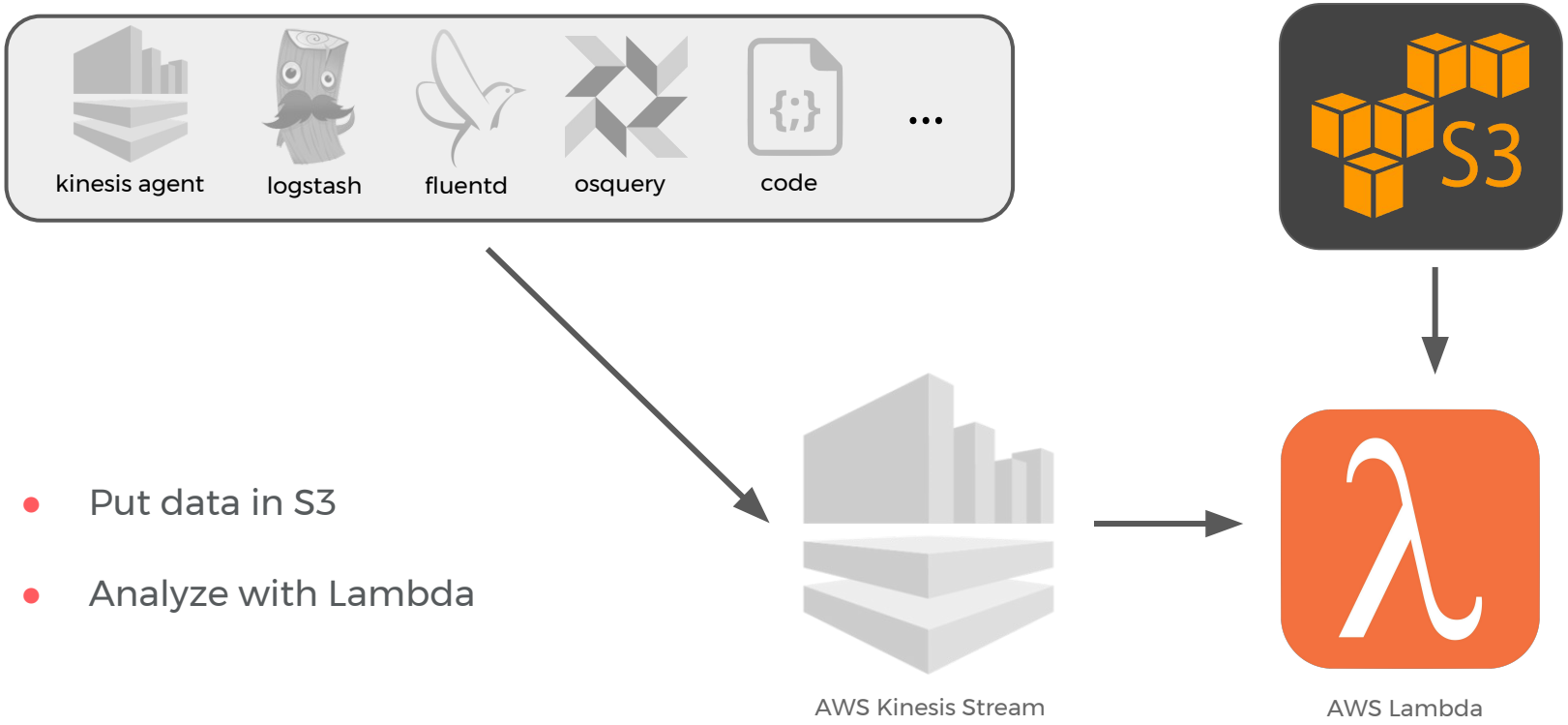


- Configure Agent
- Send to Stream
- Analyze with Lambda

AWS Kinesis Stream

AWS Lambda

Sending Data with S3



- Put data in S3
- Analyze with Lambda

Kinesis or **S3**
as a data source

- Records <= 1MB
- Performant push model

- Records > 1MB
- Less performant pull model
- Common datasource

Design

Data Analysis

Rules

Alerts

Deployment

Rules are expressed as Python functions!

Rule Layout

```
@rule(log_sources=[], match=[], outputs=[])
def rule_func(rec):
    """Description"""
    return True
```

Rule Processing Example

```
{
  "name": "logged_in_users",
  "hostIdentifier": "host1",
  "calendarTime": "Sat Dec 10 22:45:52 2016",
  "columns": {
    "host": "10.0.2.2",
    "user": "mike"
  }
}
```

Example Rule #1

```
@rule(log_sources=['osquery'], match=[], outputs=['pagerduty'])
def invalid_user(rec):
    """Catch unauthorized user logins"""
    auth_users = {'alice', 'bob'}
    query = rec['name']            # logged_in_users
    user = rec['columns']['user'] # mike

    return (
        query == 'logged_in_users' and
        user not in auth_users
    )
```

Example Rule #2

```
from netaddr import IPAddress, IPNetwork


@rule(log_sources=['osquery'], match=[], outputs=['pagerduty'])
def unauth_subnet(rec):
    """Catch logins from unauthorized subnets"""
    query = rec['name']
    ip = IPAddress(rec['columns']['host']) # 10.0.2.2
    valid_cidr = IPNetwork('10.2.0.0/24')


    return (
        query == 'logged_in_users' and
        ip not in valid_cidr
    )
```

Let’s reduce some repeated code with a ‘matcher’

Matcher

```
@matcher()
def logged_in_users(rec):
    query = rec['name']
    return query == 'logged_in_users'
```

```
from netaddr import IPAddress, IPNetwork

@rule(log_sources=['osquery'],
match=['logged_in_users'], outputs=['pagerduty'])
def invalid_subnet(rec):
    """Catch logins from unauthorized subnets"""
    ip = IPAddress(rec['columns']['host'])
    valid_cidr = IPNetwork('10.2.0.0/24')

    return ip not in valid_cidr
```

Matchers can also be used for determining:

- Environments
- Roles
- System Platforms

Design

Data Analysis

Rules

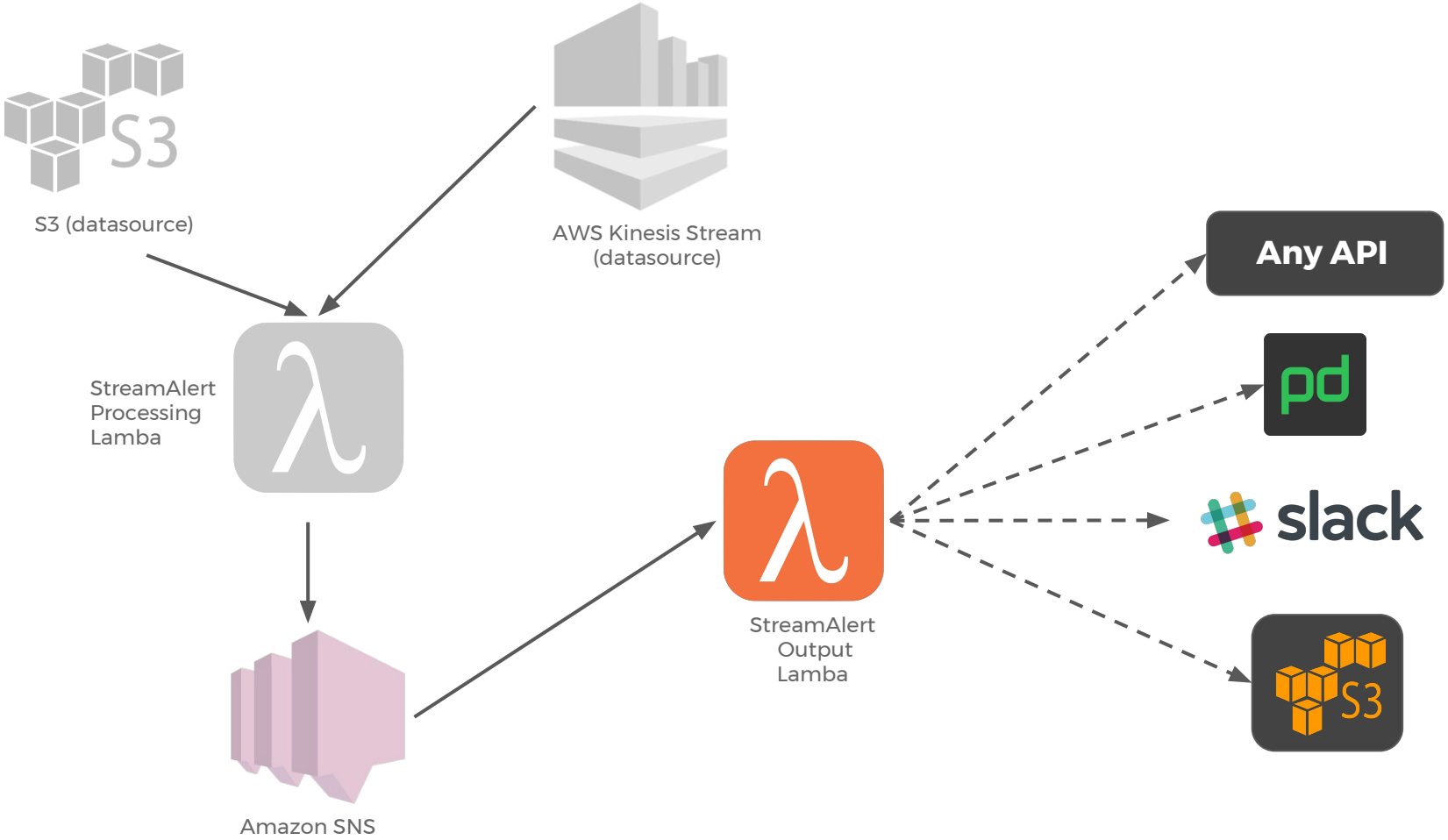
Alerts

Deployment

Alert Output Configuration

```
@rule(log_sources=['osquery'],
match=['logged_in_users'], outputs=['pagerduty'])
def invalid_subnet(rec):
    """Catch logins from unauthorized subnets"""
    ip = IPAddress(rec['columns']['host'])
    valid_cidr = IPNetwork('10.2.0.0/24')

    return ip not in valid_cidr
```


S3
S3 (datasource)
AWS Kinesis Stream
(datasource)
Any API
pd
StreamAlert
Processing
Lamba
slack
StreamAlert
Output
Lamba
S3
Amazon SNS

INCIDENTS > INCIDENT DETAILS

#38232: StreamAlert Rule Triggered - demo_invalid_login

Edit


```
rule_name    demo_invalid_login
payload
    service    kinesis
    record
        unixTime        1470824034
        name            last
        hostIdentifier  demo.host.net
        columns
            username    joebob
            type        7
            tty         pts/0
            time        12345678
            pid         139
            host        10.0.0.2
```

StreamAlert BOT 2:42 PM

StreamAlert Rule Triggered

Rule	# of Alerts
demo_invalid_login	2
Service	**Entity**
kinesis	demo_kinsis_stream

Today at 2:42 PM


```
{
    "action": "added",
    "calendarTime": "Jan 10 2017",
    "columns": {
        "host": "10.0.0.2",
        "pid": "139",
        "time": "12345678",
        "tty": "pts/0",
        "type": "7",
        "username": "joebob"
    },
    "decorations": {
        "envIdentifier": "demo",
        "roleIdentifier": "demo"
    },
    "hostIdentifier": "demo.host.net",
    "name": "last",
    "unixTime": "1470824034"
}
```

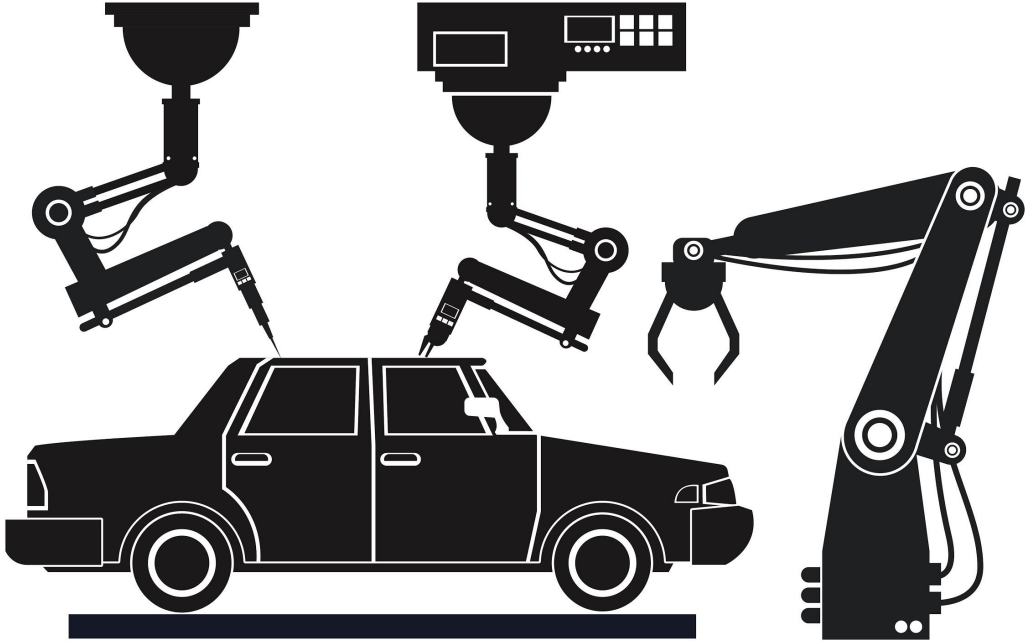
Design

Data Analysis

Rules and Alerts

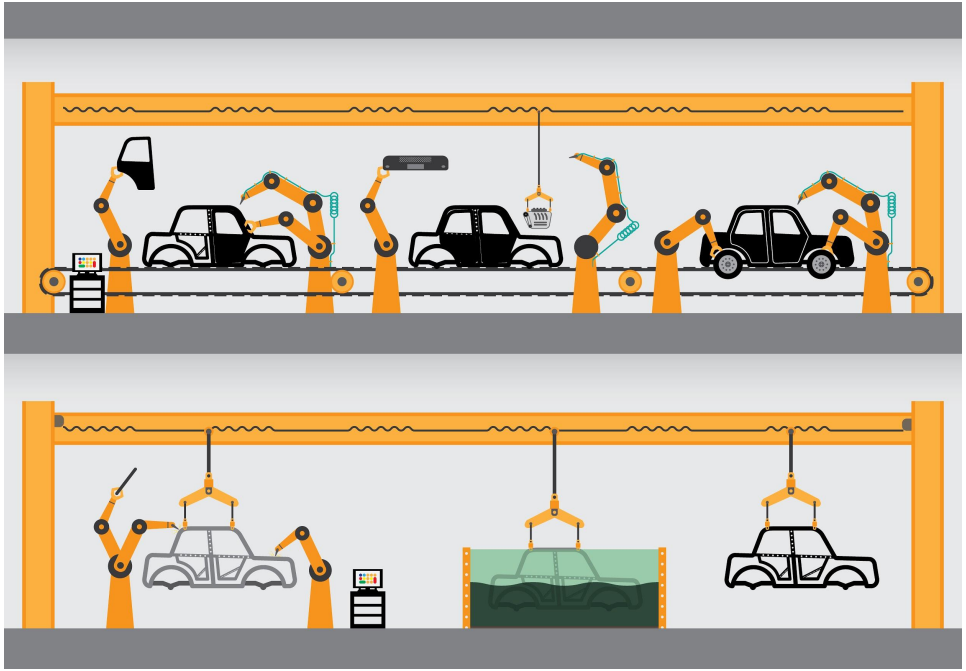
Deployment

Goal: Make Deployment Simple



Assembly Line

- Time/Cost Savings
- Accessible
- Interchangeable
- Repeatable



Building with Terraform

- Express complex infrastructure as code
- Interchangeable
- Consistent
- Abstracted with stream_alert_cli

web : github.com/airbnb/streamalert

twitter: @streamalert_io

Thank You!

- **@enigmaconf, @usenix**
- **@awscloud team** (services and support)

- **@mimeframe** (concept, website, guides, review)
- **@strcrzy** (core rules logic)
- **@zwass** (osquery kinesis output plugins)
- **@hackgnar** (osquery kinesis bug fixes)

airbnb