ERLANG MICRO-SERVICES WITH ALL THE BUZZWORDS

Chad Gibbons
Sr. Director, Security Engineering
Erlang User Conference 2017

ALERT LOGIC

What's this all about?

Micro-service

REST APIs

Pub/Sub Messaging

Cloud

DevOps

Continuous Integration and Deployment

Software-as-a-Service

Multi-tenant

Big Data

CONTEXT SETTING

Who is Alert Logic?

Security-as-a-Service Solution

- Monitor and Ingest customer data – lots of it
- Analyze and Detect Security Vulnerabilities and Incidents
- Security Operations Center expert analysis and guidance

Alert Logic Engineering History

Early Days: 2002 - 2005

- Startup / Integration Mode
- Database-focused integration

Growing up – 2005 - 2011

- Log Management feature added
- Highly scalable data ingestion and search platform

Expansion – 2011 - 2013

- Cloud explosion
- Services-based applications

We Wanted a New Approach

Dramatically increase quality and capabilities

- Provide an architectural foundation for everything we build
- Define a new engineering culture

Starting Over

Distributed, micro-services architecture

Focus on the interfaces: HTTP APIs and pub/sub messaging

Recognize Conway's Law: let teams be small, focused, and responsible for their work

Mandate as little as possible; encourage and make the best path *easy*

Document and follow a set of design principles and use best practices

DESIGN PRINCIPLES

APIs

Everything is an API

- Every service provides a REST API for integration and monitoring
- Canonical API paths
  - https://*<public-api-endpoint>*/*<service-name>*/*<API-version>*/*[account-ID]*/*<resource>*
  - https://api.*example*.alertlogic.com/aims/v1/67000001/users

Every API is Public

Every API is considered public by default

No backdoor APIs for our User Interfaces

API Documentation and consistency considered best practice for every service

API Documentation within the Product UI

Getting Started with Alert Logic

Getting Started with API Documentation

AIMS
Access and Identity Management Service

Assets
Assets management service.

Azure Explorer
Microsoft Azure Cloud discovery service.

Cargo
Report delivery service.

Cloud Explorer
Environment discovery and monitoring service.

Credentials
Host scanning credentials repository.

Dashboards
UI dashboards service.

Environments
Environments service.

Informant
Activity log monitor service.

Inquisitor
Static Analysis Engine for Cloud Environment Configuration.

Launcher
Alert Logic security infrastructure deployment service.

Scan Results
Raw scan results storage service.

Scan Scheduler
Scan scheduling service.

Sources
Sources service.

Strawboss
Task and task list manager service.

Tacoma
Tableau content manager.

Ticket Master
Tableau identity bridge service.

Usage
Usage service.

Vulnerabilities
Vulnerabilities knowledgebase service.

Watchlist
Assets bookmarking service.

API Documentation Example

ALERT LOGIC®
Security. Compliance. Cloud.

Trusted Tickets

ticketmaster

Trusted Tickets

Create a trusted ticket with Tableau

Trusted Tickets - Create a trusted ticket with Tableau

1.0.0

Creates a trusted ticket with Tableau for the user specified by the AIMS authentication token provided in the call

POST

```
/ticketmaster/v1/:account_id/ticket
```

Create Ticket:

```
curl -X POST -H'x-aims-auth-token:...' https://api.route105.net/ticketmaster/v1/67000001/ticket
```

Header

Field	Type	Description
x-aims-auth-token	String	AuthenticationToken returned by AIMS service.

ALERT LOGIC®

Pervasive AAA

Pervasive Authentication, Authorization, and Auditing

- **ALL** API calls are authenticated, authorized, and audited
- Provided by the service framework software layer
- Permission strings defined within the services themselves
  - *service:[account-ID]:operation:object*
- Every user, and every service, has its own identity

Example Permissions

```
%%-----------------------------------------------------
%% ticketmaster service permissions
%%
required_permission(post, [AccountId, <<"ticket">>], _Req) ->
    <<"ticketmaster:", AccountId/binary, ":create:ticket">>.

%%-----------------------------------------------------
%% otto service permissions
%%
required_permission(get, [<<"deployment">>], _) ->
    <<"otto::view:deployment">>;

required_permission(post,  [<<"deployment">>], _) ->
    <<"otto::manage:deployment">>;
```

No Web Server

There **is no** web application server

- JavaScript-based UI
- Content provided by CDN (AWS CloudFront) and not a web server
- **No** business rules within the UI
- **Only** public API access for the UI

Automated Deployment

100% automated deployment in AWS, of **100%** of the environment

- AWS CloudFormation used as a basis for everything
- **No** shortcuts

Service CloudFormation

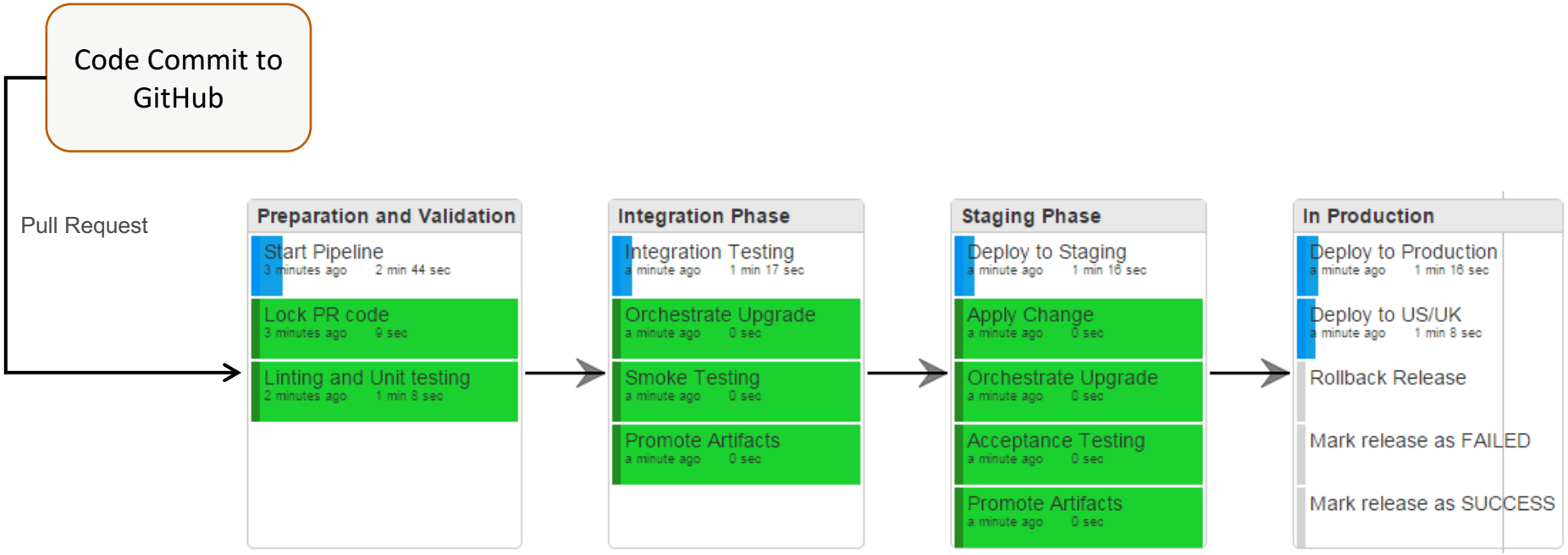
```
"cfnStackTicketmaster": {
      "service": "ticketmaster",
      "ami_version": "ticketmaster/alertlogic/v1.4.1",
      "depends_on": [
        "cfnStackRabbitMQ",
        "cfnStackAIMS",
        "cfnStackTableau"
      ],
      "security_groups": [
        "cfnStackRabbitMQ.sgRabbitMQClient",
        "cfnStackTableau.sgTableauClient"
      ],
      "iam_role": "cfnStackIam.iamRoleBackendServer",
      "iam_profile": "cfnStackIam.iamInstanceProfileBackendServer"
    }
```

Continuous Deployment

Release small, testable, **loosely-coupled** components into production

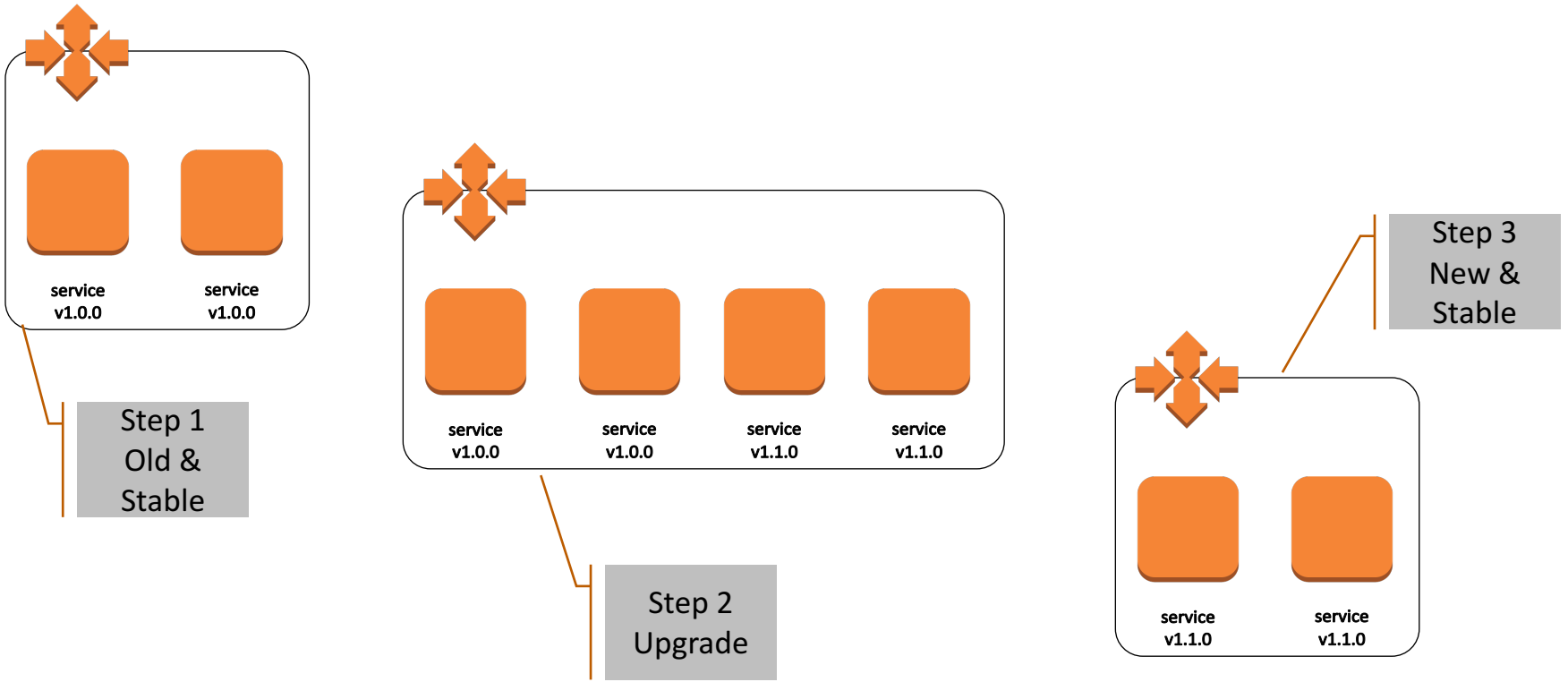
- One of the most positive improvements I've seen in my career

Deployment Pipeline Release Lifecycle



ALERT LOGIC

Service Upgrades



Infrastructure

Avoid operating custom infrastructure

- Leverage AWS services when possible
- Running our own infrastructure not cost effective nor a key competency

Minimize Configuration

Minimize or **eliminate** configuration

- Design services to self-configure and learn from the environment
- Service Discovery!

Service Discovery

Services Find Each Other

- Dynamic service end-points

```
$ al-sd-get-service-endpoints aws/prod ticketmaster
10.0.0.112:8521
10.0.3.238:8521
```

Log Data Mutations

Log every time something in the system changes

- Leverage Kinesis to record every time a resource changes or a service event occurs
- Publish state changes to message bus

Dynamic Scalability

Scale dynamically and manage services per-customer

- API paths include customer account IDs, allowing intelligent routing of calls to specific service instances

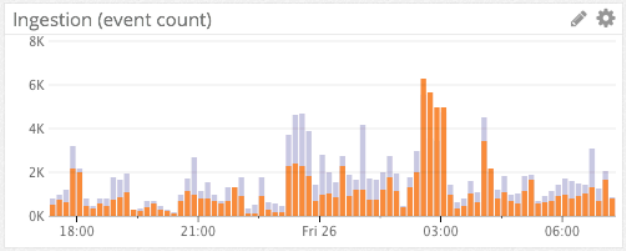
- Shared-nothing services preferred for easy auto-scaling

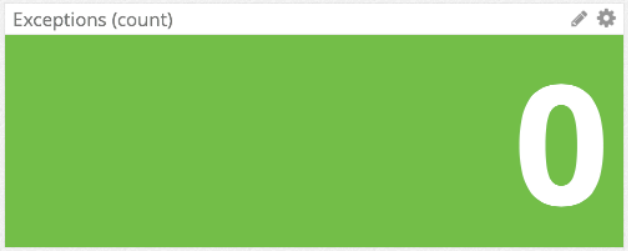
Metrics and Monitoring

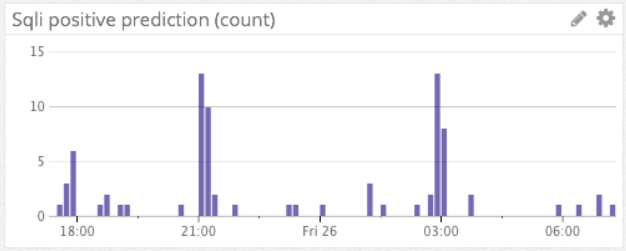
Constantly evaluate service stability, availability, and performance

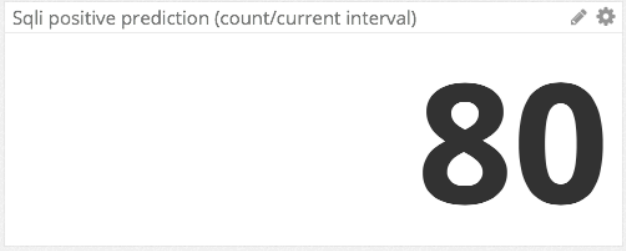
- Development team review of metrics key
- Metrics and monitoring becomes part of the engineering lifecycle

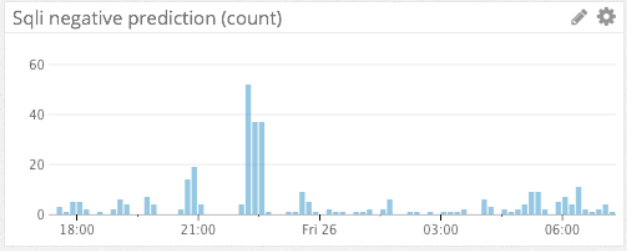
DevOps-Focused Dashboards

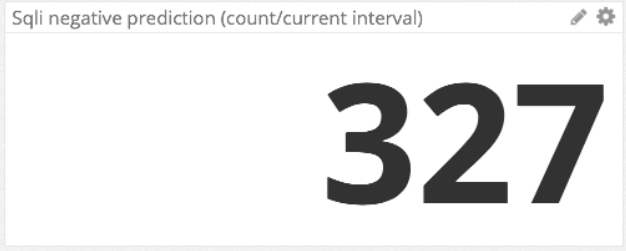


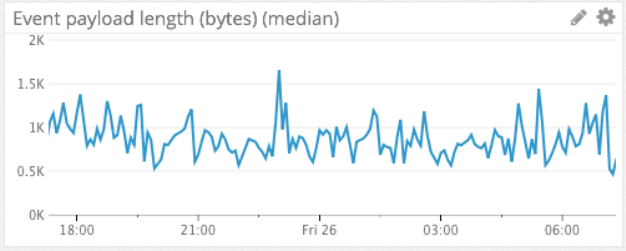


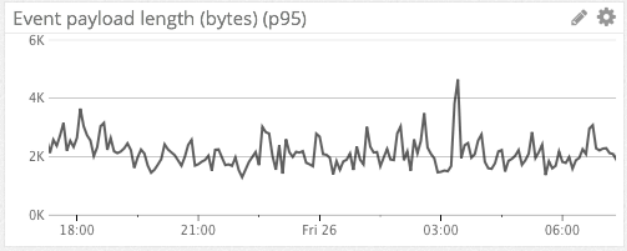


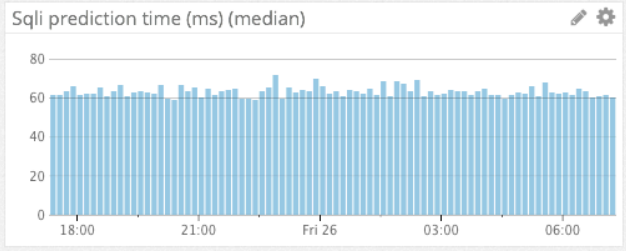












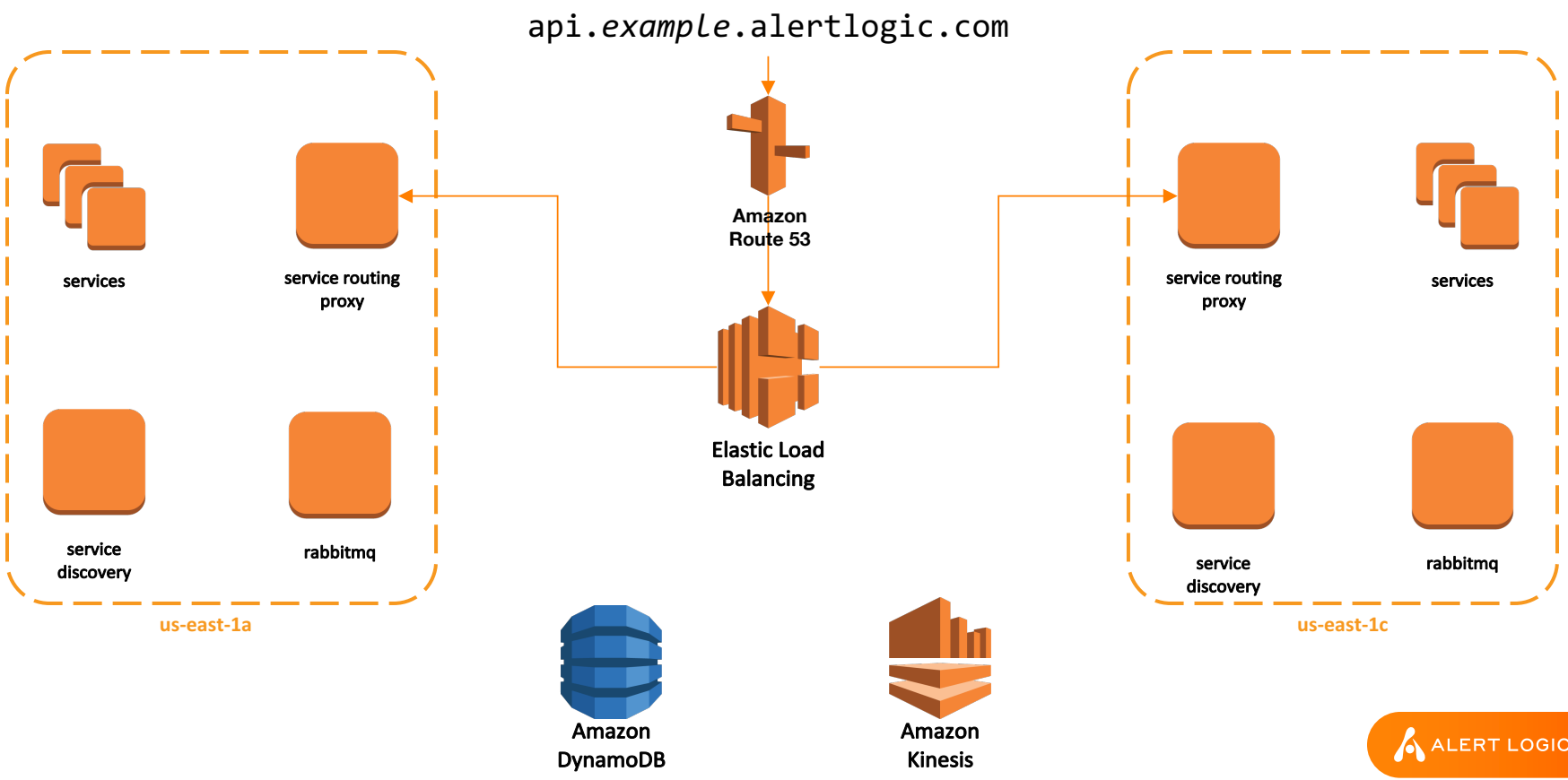
ALERT LOGIC

Ownership Culture

Focused teams with long-term **ownership** of development, test, and production

REAL-WORLD

Deployment Architecture



ALERT LOGIC

Lessons Learned – Service Discovery

Service Discovery is hard!

- Avoid doing this yourself
- Leverage existing solutions when possible, such as Netflix's Eureka

Lessons Learned - AWS

High-availability and Disaster Recovery must be designed into every system

AWS Cost Management is an Engineering Requirement

Use Containers!

Lessons Learned - Service Composition

How big should micro-services be?

- We settled for services that own a specific data resource
- Composite services a necessity as the system grows

Lessons Learned - Culture

Great culture doesn’t happen without effort

Cultural and Engineering change is politics – don’t avoid it

Lessons Learned - Erlang

What about Erlang?

- A great choice for services
- But, community support around many libraries minimal
- AWS library support provided by https://github.com/erlcloud/erlcloud
  - Help out!

WRAP UP

Alert Logic Locations

UNITED STATES

Houston, TX - CORPORATE HEADQUARTERS

Seattle, WA

Dallas, TX

Austin, TX

EUROPE

Cardiff - EUROPEAN HEADQUARTERS

Guildford - UK SALES & MARKETING

Belfast

COLOMBIA

Cali, Colombia

ALERT LOGIC

Want More Information?

Company Website: http://www.alertlogic.com/

E-mail: cgibbons@alertlogic.com
dcgibbons@gmail.com

LinkedIn: https://www.linkedin.com/in/dcgibbons/

Twitter: @dcgibbons

ALERT LOGIC

Thank you.