Security In The Land of Microservices

Jack Mannino

Who Am I?

- Jack Mannino
  - CEO @ nVisium
  - Banned from computers by his co-workers
  - Does most of his development in Scala
  - Has a love/hate relationship with microservices
    - From experience building them

What's This All About?

- Microservice pattern and what's unique
- Asset and Data Inventory
- Authentication
- Access Control and Identity Management
- Securely Sharing Secrets
- ~~RASP, Top 10 lists~~

What Are Microservices?

The term "Microservice Architecture" has sprung up over the last few years to describe a particular way of designing software applications as suites of independently deployable services. While there is no precise definition of this architectural style, there are certain common characteristics around organization around business capability, automated deployment, intelligence in the endpoints, and decentralized control of languages and data. - *Martin Fowler*

Properties of Microservices

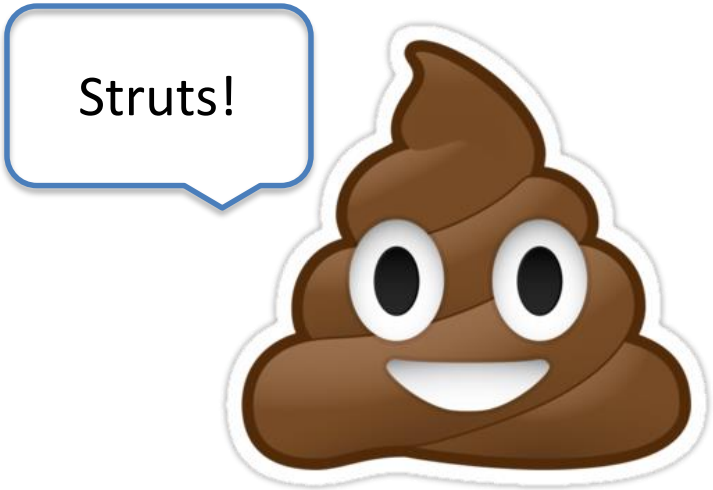
- ✓ Independently deployable services
- ✓ Decentralized management and governance
- ✓ Structured around business capabilities
- ✓ Resilient to failures and contagion between services
- ✓ “Smart endpoints, dumb pipes”

What Are Microservices?

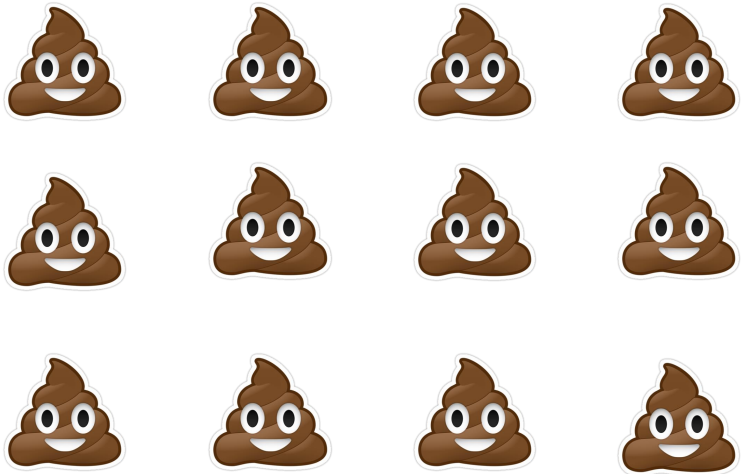
Microservices simplify everything

Wrong

SOA/ESB For Hipsters

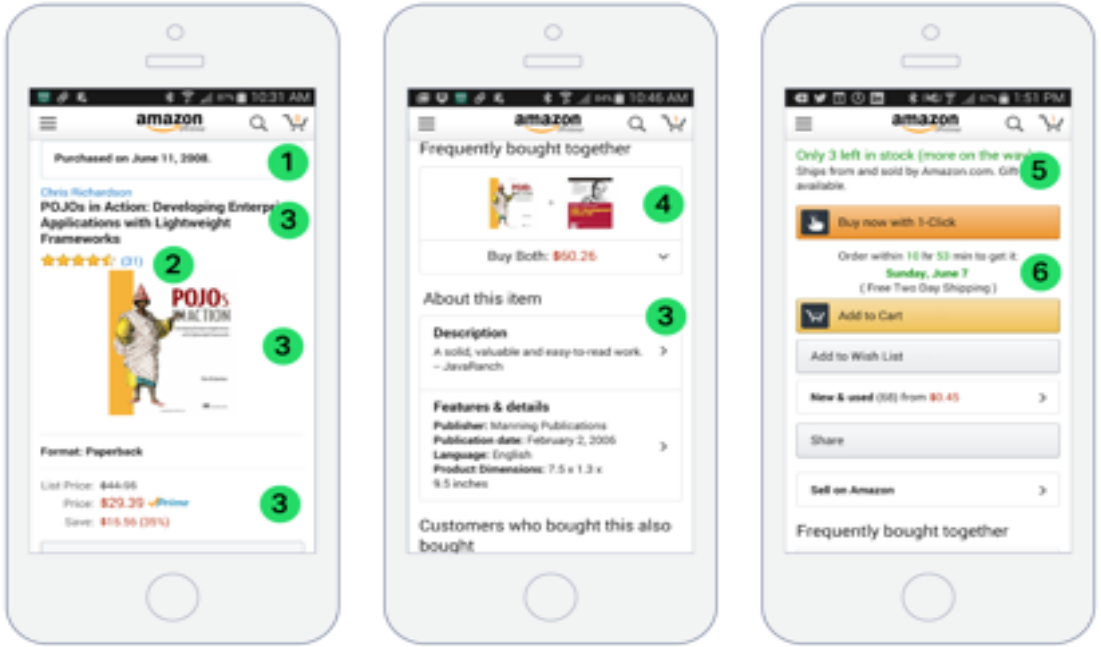


Monolithic



Microservices

One External View, Many Services



Order History

Reviews

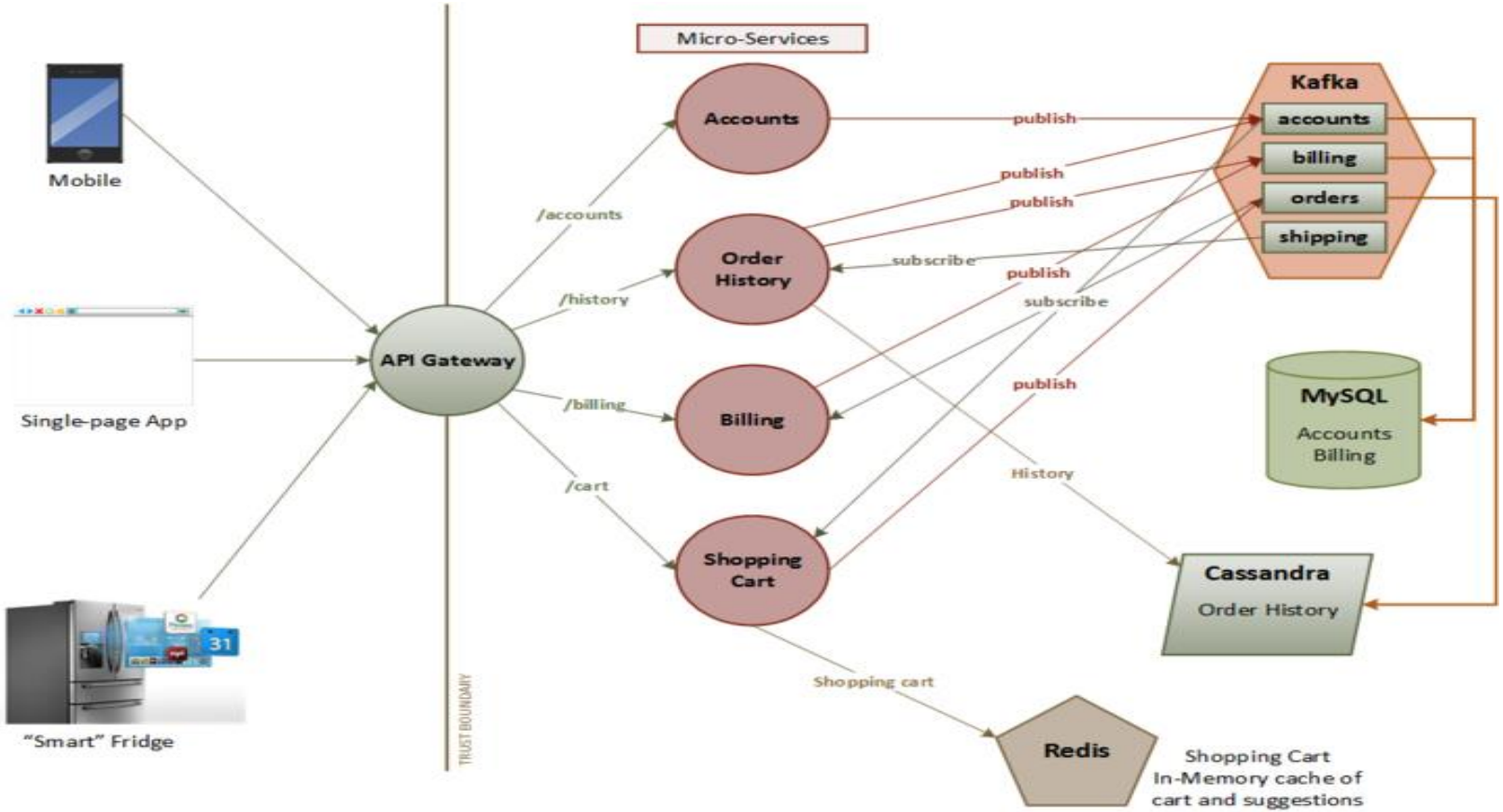
Product Information

Recommendations

Inventory

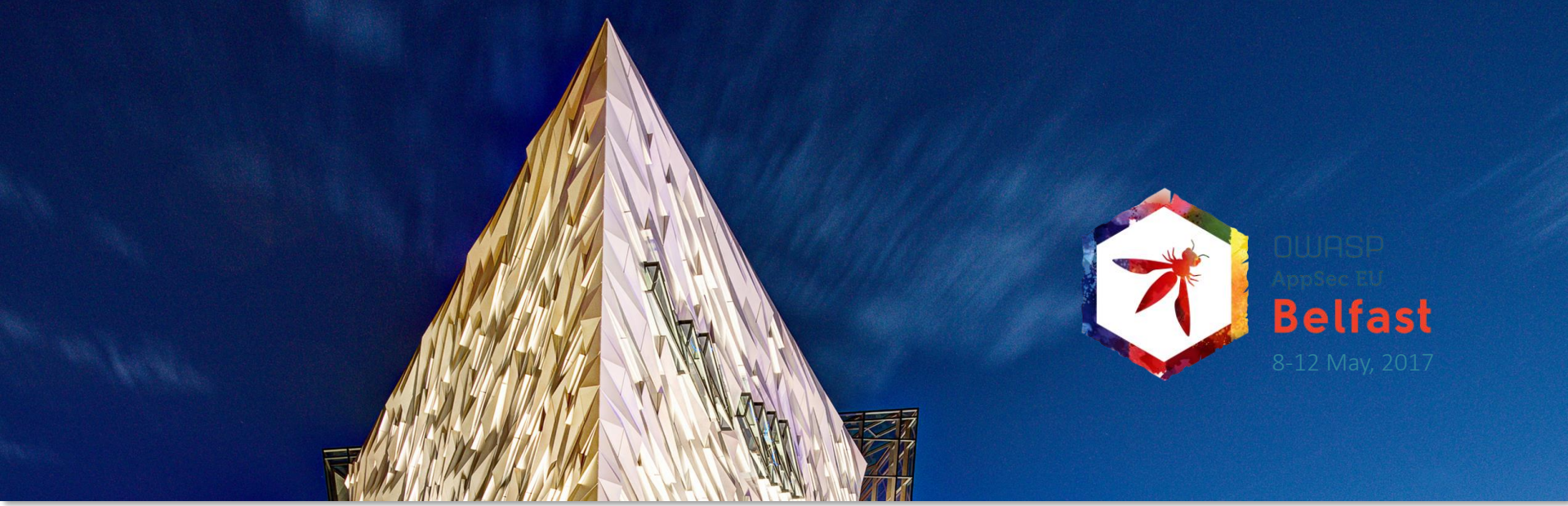
Shipping

A Simple Architecture



Step #1 – Secure Your APIs

- Your APIs are the gateway into the microservice architecture
- **Anyone selling you Cross Site Microservice Injection prevention, is lying**
- Issues like SQL Injection is still SQL Injection, but we lose source/sink visibility

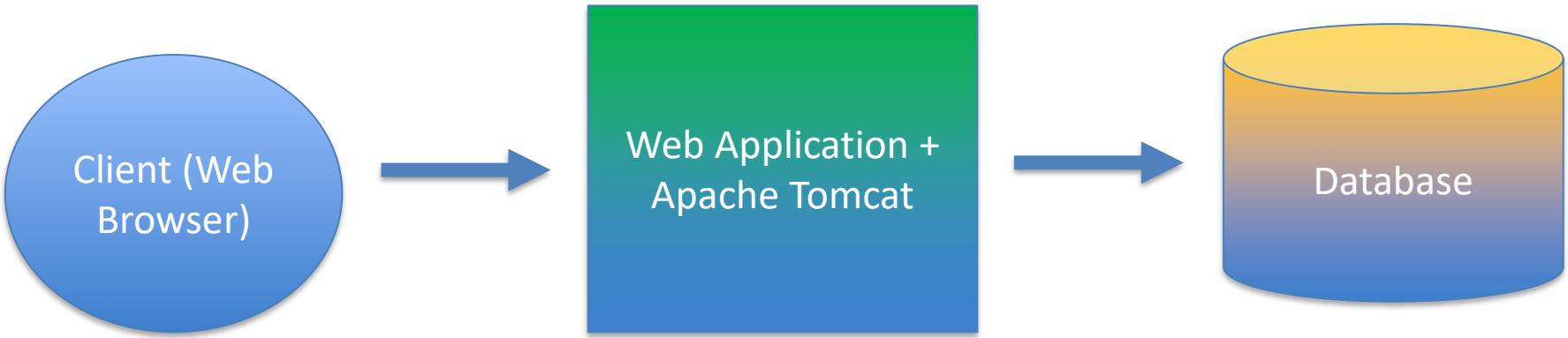


Data and Asset Inventory

Deploying Code All Day, Every Day

- Once upon a time, we released 3-4 times a year
- CI/CD, container orchestration, and Platform-as-a-Service (PaaS) has changed that

When Life Was Easy



Infrastructure-As-Code

- Code is now infrastructure
- Developer laptops often hold the keys to the kingdom
- Fairly new-ish territory for many security teams
  - Immature organization practices = massive business disruption

Infrastructure-As-Code


```
{
  "AWSTemplateFormatVersion" : "2010-09-09",

  "Description" : "DC/OS AWS CloudFormation Template",

  "Metadata": {
    "DcosImageCommit": "unset",
    "TemplateGenerationDate": "unset"
  },

  "Parameters" : {
    "KeyName" : {
      "Description" : "Required: Specify your AWS EC2 Key Pair.",
      "Type" : "AWS::EC2::KeyPair::KeyName"
    },
    "AdminLocation": {
      "Description" : "Optional: Specify the IP range to whitelist for access to the admin zone. Must be a valid CIDR.",
      "Type" : "String",
      "MinLength" : "9",
      "MaxLength" : "18",
      "Default" : "0.0.0.0/0",
      "AllowedPattern" : "^([0-9]+\\.){3}[0-9]+\\/[0-9]+$",
      "ConstraintDescription" : "must be a valid CIDR."
    },
    "SlaveInstanceCount" : {
      "Description" : "Required: Specify the number of private agent nodes or accept the default.",
      "Type" : "Number",
      "Default" : "{{ num_private_slaves }}"
    },
    "PublicSlaveInstanceCount" : {
      "Description" : "Required: Specify the number of public agent nodes or accept the default.",
      "Type" : "Number",
      "Default" : "{{ num_public_slaves }}"
```

```
provider "aws" {
  region = "${var.region}"
}
module "vpc" {
  source = "./vpc"
  key_name = "${var.key_name}"
  ip_range = "${var.ip_range}"
}
module "elb" {
  source = "./elb"
  public-subnet-az-1a = "${module.vpc.public-subnet-az-1a}"
  public-subnet-az-1b = "${module.vpc.public-subnet-az-1b}"
  elb_http_inbound_sg_id = "${module.vpc.elb_http_inbound_sg_id}"
}
module "asg" {
  source ="./asg"
  public-subnet-az-1a = "${module.vpc.public-subnet-az-1a}"
  public-subnet-az-1b = "${module.vpc.public-subnet-az-1b}"
  dev-http-lc-id = "${module.lc.dev-http-lc-id}"
  dev-http-lc-name = "${module.lc.dev-http-lc-name}"
  dev-web-elb-name = "${module.elb.dev_web_elb_name}"
}
module "lc" {
  source = "./lc"
  ec2_http_inbound_sg_id = "${module.vpc.ec2_http_inbound_sg_id}"
  key_name = "${var.key_name}"
}
```

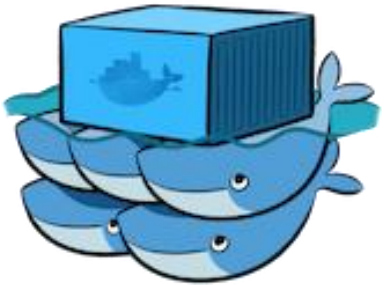
Infrastructure-As-Code

- Now, your architecture might be in a GitHub repo
- Important to restrict who can commit to master
- Important to review code merges (pull requests, etc)
- Great for auditability and inventory management, if done correctly

Containers and Orchestration

kubernetes
by Google

DC/OS



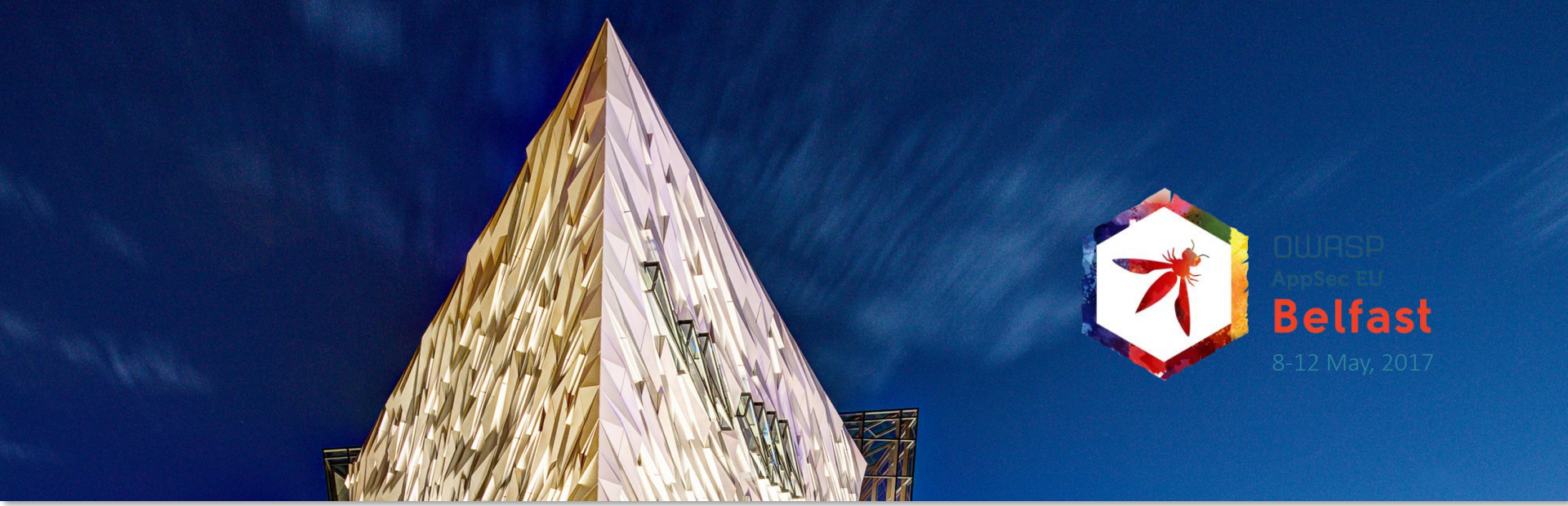
OPENSHIFT

Serverless Functions

- Tools like AWS Lambda, Azure Cloud Functions
- Stateless and short-lived
- Finish your work in 5 minutes or “die”
- Monorepos vs. distributed repositories
- Security tools with performance hit?
  - ***Enjoy getting laughed at***

Where's My Data? Clean Up Your Toys

- Intentional persistence
- Message queues and commit logs
- Code repositories
- Developer laptops
- Zeppelin notebooks
- Everywhere?

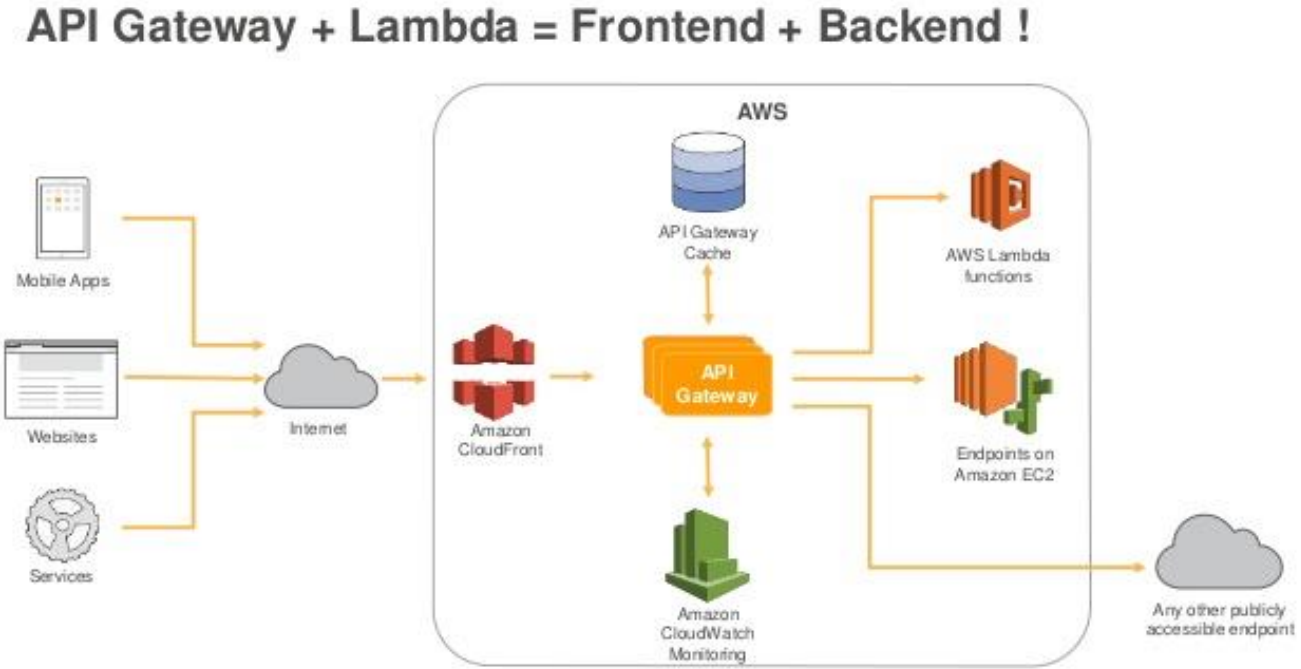


Authentication

API Gateway Pattern

- API Gateway is the most prolific Microservice authentication pattern
- Similar to the Façade pattern
- Encapsulates internal architecture
- Abstracts your services from authentication
- Many implementations
  - AWS API Gateway
  - Azure API Gateway
  - Mashape Kong

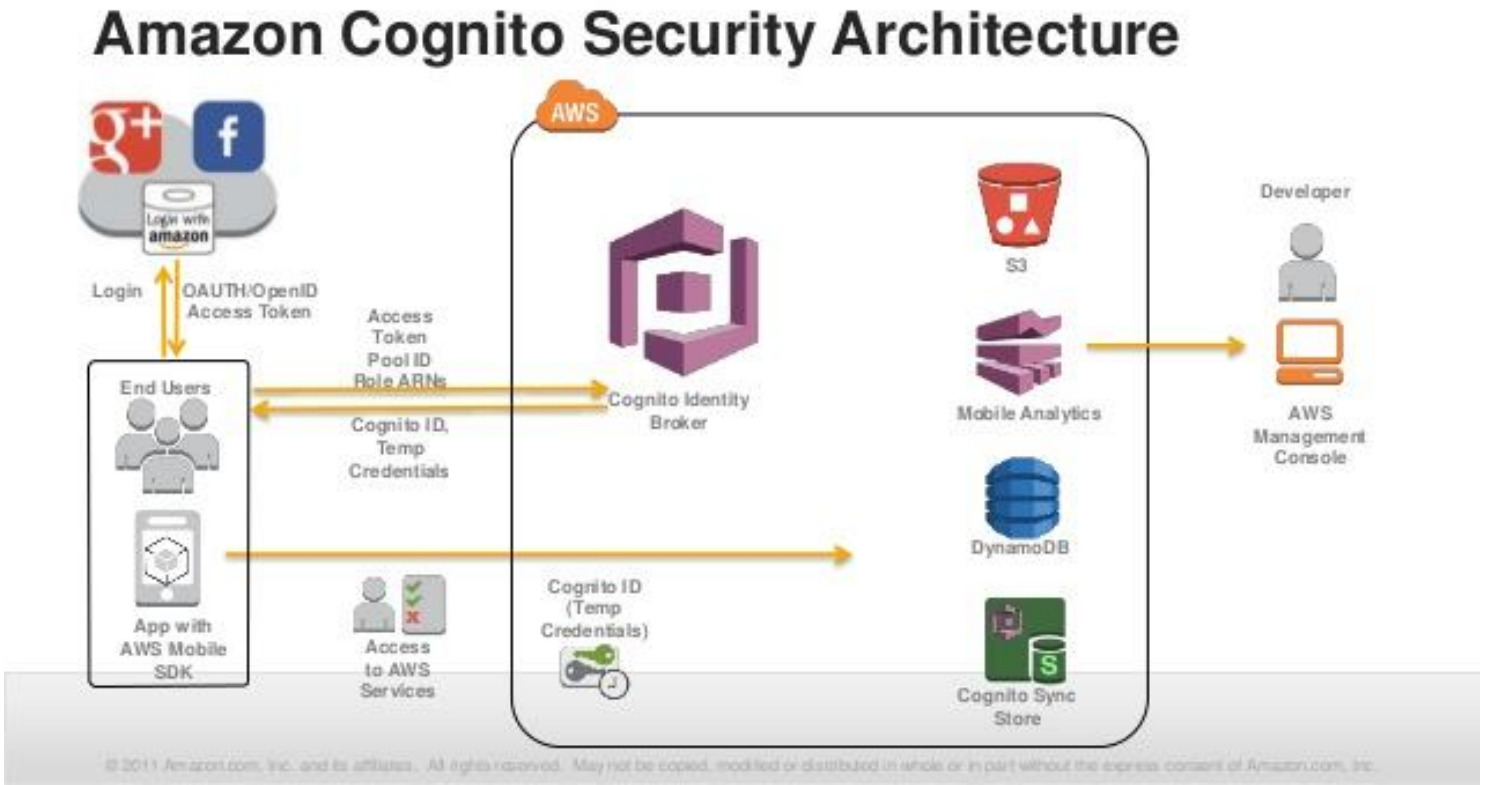
API Gateway Pattern



API Gateway Pattern

- At my day job, we use AWS API Gateway + Cognito to handle the heavy lifting
  - https://aws.amazon.com/blogs/mobile/integrating-amazon-cognito-user-pools-with-api-gateway/
  - Cognito handles authenticating with credentials, MFA, etc.

API Gateway + Cognito

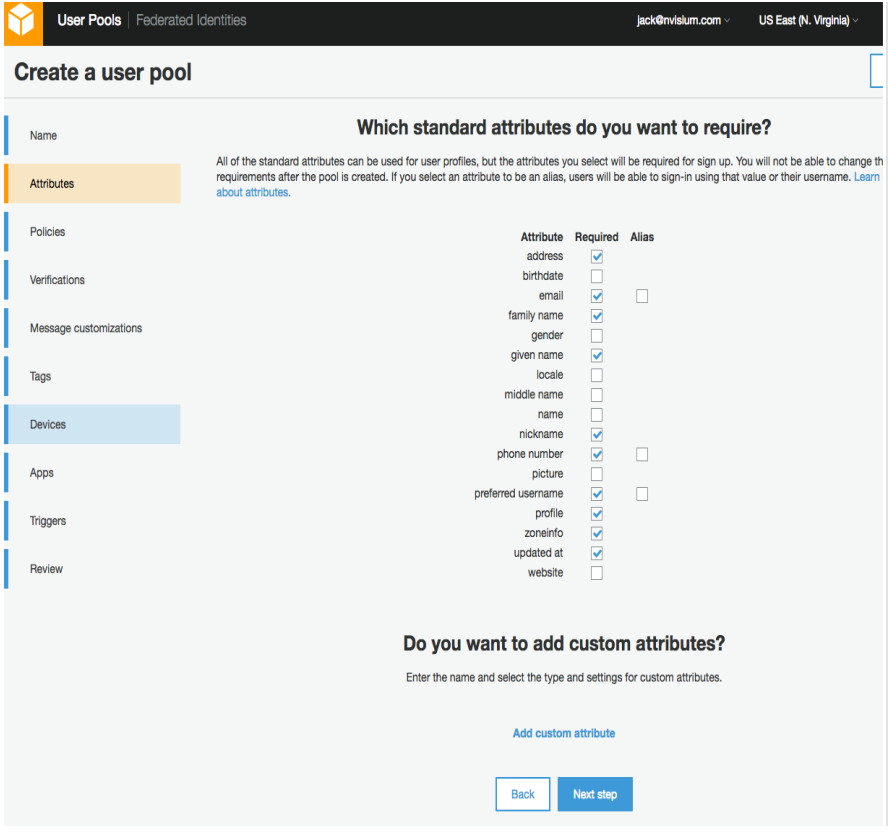


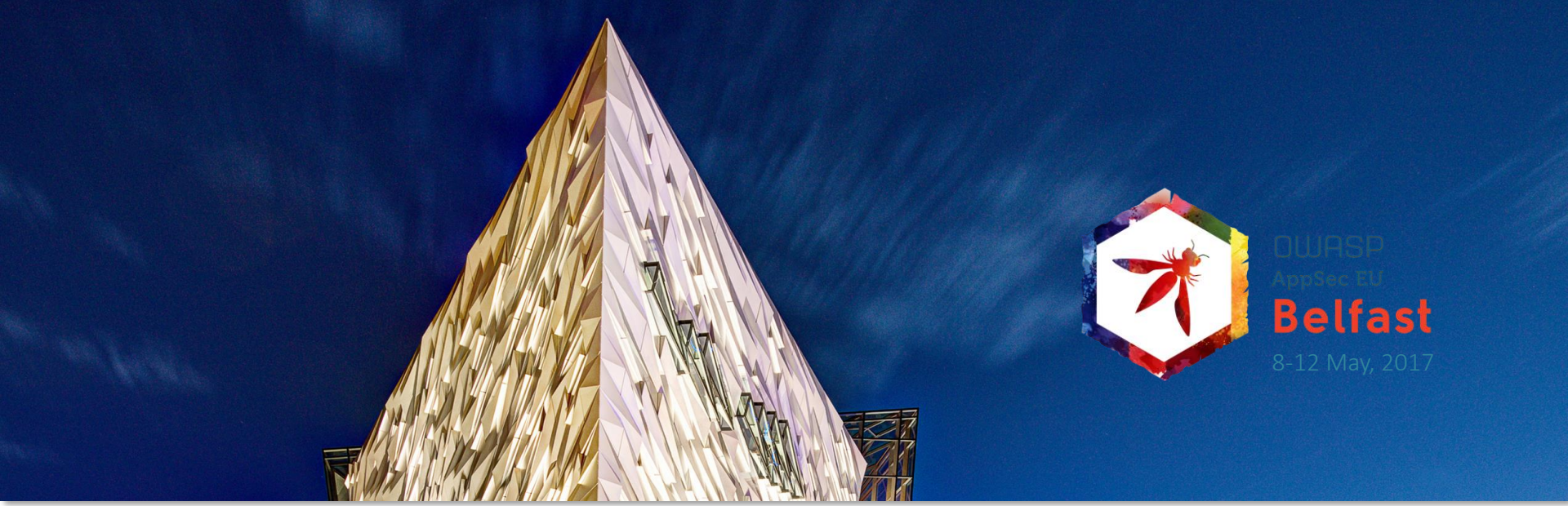
API Gateway Pattern

- Each request is signed, which provides an additional layer of authentication
  - https://docs.aws.amazon.com/apigateway/api-reference/signing-requests/
  - Integrate Lambda functions for pre/post processing hooks
  - Bonus: good architecture = breaks CSRF if done correctly
- Can consume your Swagger files

Once You Get Past The Gateway

- The gateway can share data with downstream services
- Lambda function post-processing
- Standard + Custom attributes
  - Username, email
  - Custom attributes for your app





Access Control and Identity Management

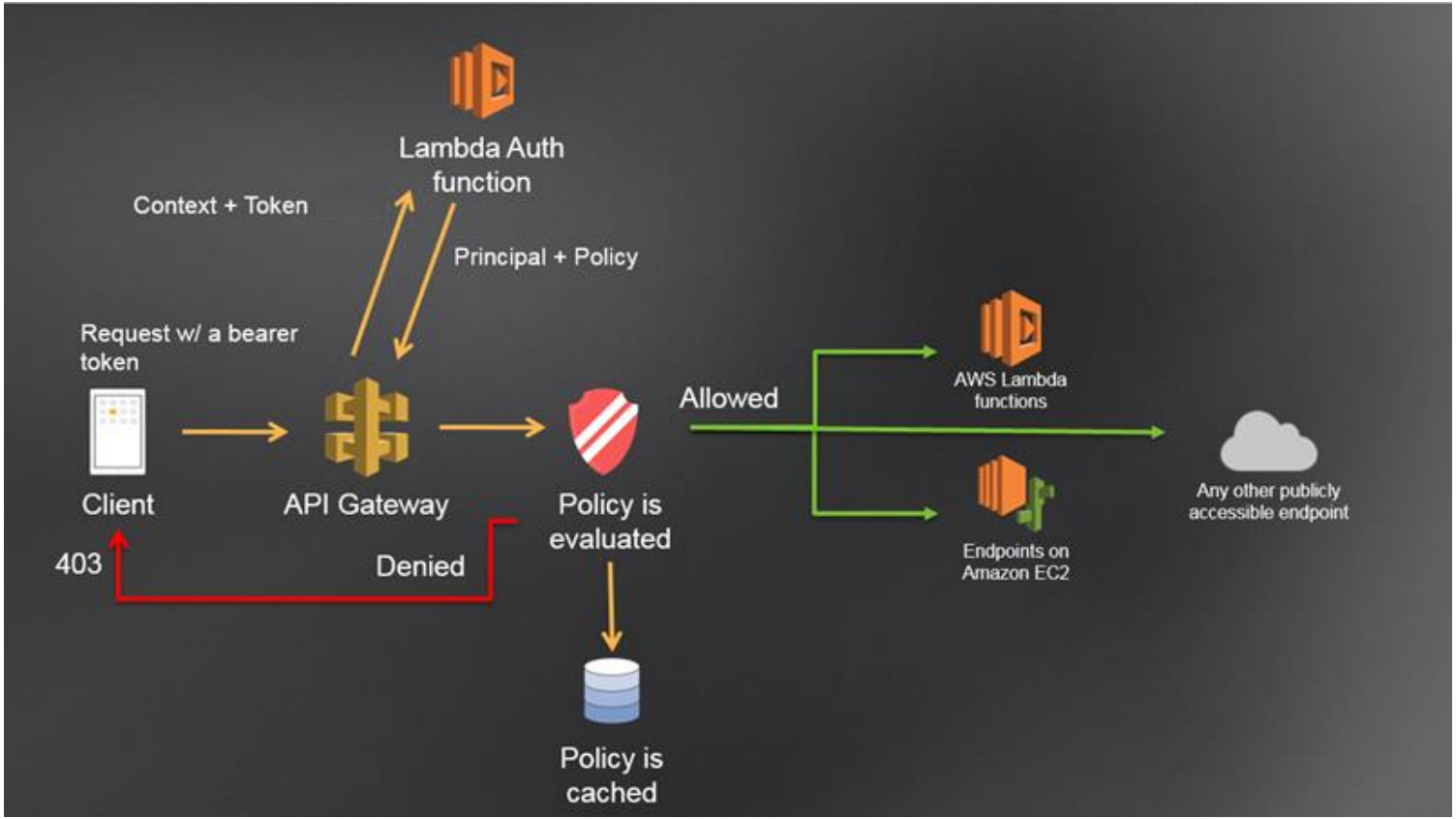
Decentralized Sanity

- So we've decentralized things, right?
- API Gateway + JWT can help
- Patterns like CQRS can architecturally limit damage and exposure across interfaces

JSON Web Tokens (JWT)

- Love it or hate it, JWT allows us to pass identity and claims across services
- We still need to consider the business rules of each service, but we have a starting point

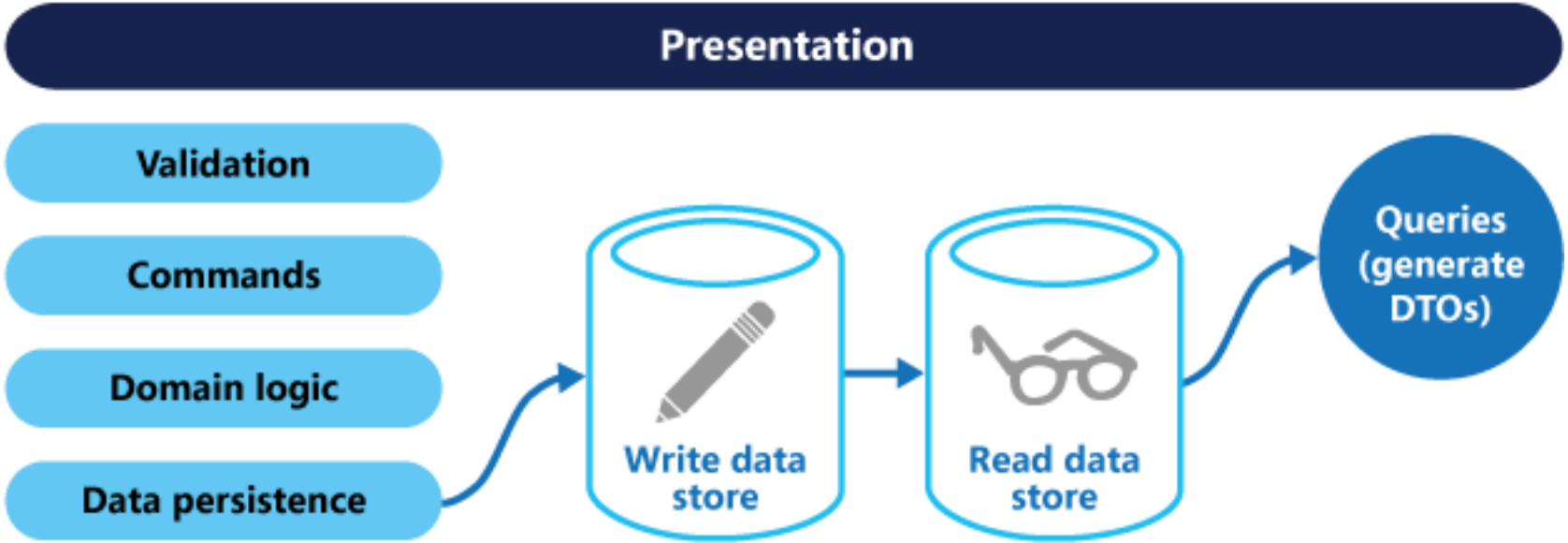
JSON Web Tokens (JWT)



Command Query Responsibility Segregation (CQRS)

- Command and query interfaces are separated
- Independent read and write models
- Possibly, independent data stores for read and write
- We have a lot more granular control over which services and users we authorize around capabilities

Command Query Responsibility Segregation (CQRS)



What About Between Services?

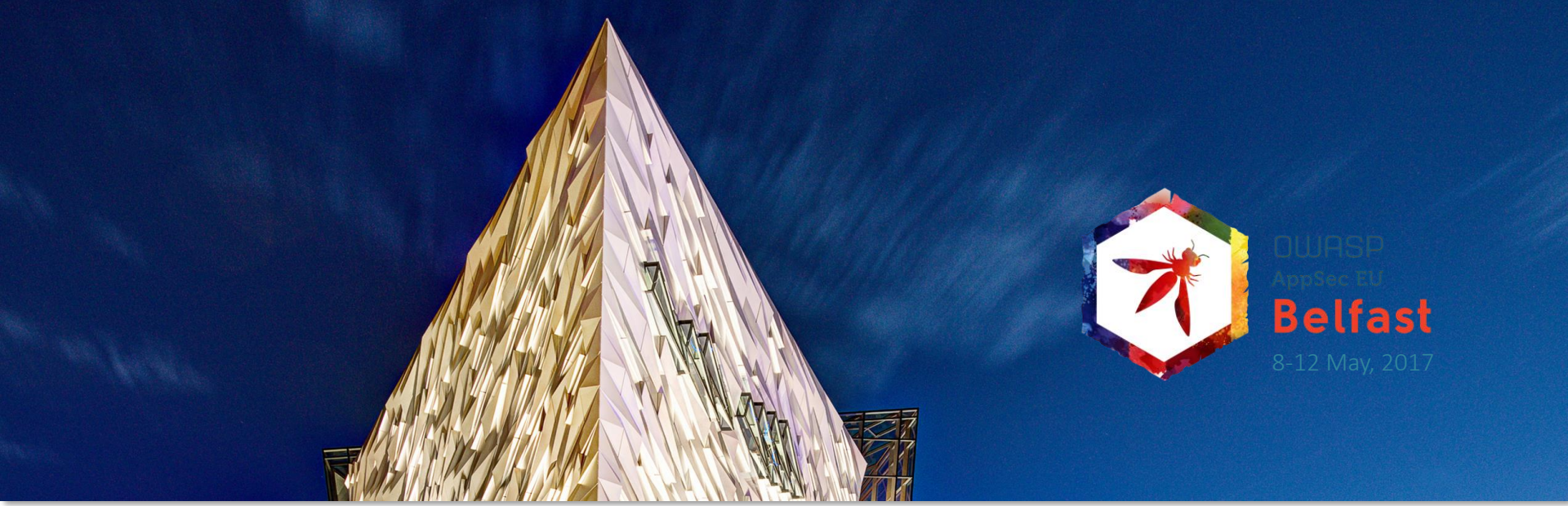
- Think of scenarios like messaging between services
  - Identify trusted publishers and subscribers
  - Determine which services should consume your data, and limit scope
  - Apply authentication and topic-level authorization

Producer:

kafka-console-producer.sh --broker-list localhost:9092 --topic creditcard-stuff
This is a credit card # 1234567890123456
This is a credit card # 1234567890111111

Consumer:

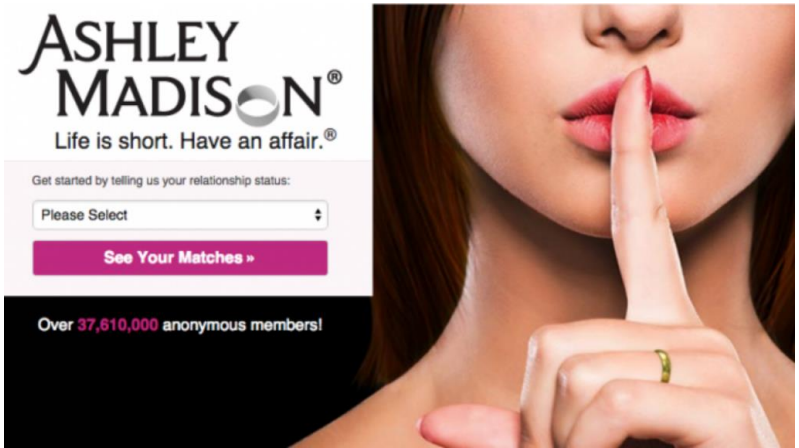
kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic creditcard-stuff --from-beginning



Securely Sharing Secrets

Keeping Secrets

- You don’t want to pass credentials around in plain text (no-brainer, right?)
- There are options, but they vary with mileage



Bad Ideas

- 👎 Hardcoding credentials in your code
- 👎 Hardcoding credentials in your Dockerfile
- 👎 Using environment variables to pass secrets

Hardcoded Secrets

```
# CentOS7 + ssh + ttyjs (in Supervisord)
FROM centos:centos7
RUN \
  yum update -y && \
  yum install -y epel-release && \
  yum install -y net-tools python-setuptools hostname inotify-tools yum-utils coreutils pwgen && \
  yum clean all

# Install supervisord
RUN  easy_install supervisor

# Prepare the container for our use with a Volume for logs and apps (/data)
# Prepare the container for our use with a directory to run custom bootstrap procedure
RUN mkdir -p /config/init && \
    mkdir -p /data

# Environment variable provision to accept root-password while creating container
ENV ROOT_PASS password

# Create /etc/supervisord.conf
RUN echo "[supervisord]" > /etc/supervisord.conf && \
    echo "pidfile = /run/supervisord.pid" >> /etc/supervisord.conf && \
    echo "# It seems that it's not possible to swith this log to NONE (it creates NONE logfile)" >> /etc/supervisord.conf && \
    echo "logfile = /data/logs/supervisord.log" >> /etc/supervisord.conf && \
    echo "# Set loglevel=debug, only then all logs from child services are printed out" >> /etc/supervisord.conf && \
    echo "# to container logs (and thus available via the command below - " >> /etc/supervisord.conf && \
    echo "#             docker logs [container]" >> /etc/supervisord.conf && \
    echo "loglevel = debug" >> /etc/supervisord.conf && \
    echo "" >> /etc/supervisord.conf && \
    echo "# These two (unix_http_server, rpcinterface) are needed for supervisorctl to work" >> /etc/supervisord.conf && \
    echo "[inet_http_server]" >> /etc/supervisord.conf && \
    echo "port = :9111" >> /etc/supervisord.conf && \
    echo "username = sv" >> /etc/supervisord.conf && \
    echo "password = password" >> /etc/supervisord.conf && \
```

Secrets Via Environment Variables

docker run –it –e “DBUSER=dbuser” –e “DBPASSWD=dbpasswd” mydbimage

- 👎 Every process in your container can read these variables
- 👎 You risk leaking secrets via dashboards, logs and history

A Perfect Solution?

- Not quite, but Docker leads the pack 
- Kubernetes, DC/OS, OpenShift all have options too

Passing Secrets To A Kubernetes Pod

OWASP AppSec EU Belfast

Create Files Use a Secret

```
$ echo -n "administrator" > ./username.txt
$ echo -n "0XDEADB33F" > ./password.txt
```

Create Secrets

```
$ kubectl create secret generic db-user-pw
--from-file=./username.txt --
from-file=./password.txt secret "db-user-
pw" created
```

Use A Secret

```
"apiVersion": "v1",
"kind": "Pod",
 "metadata": {
  "name": "jackpod",
  "namespace": "jack"
 },
 "spec": {
  "containers": [{
   "name": "cart-cache",
   "image": "redis",
   "volumeMounts": [{
    "name": "redis-secrets",
    "mountPath": "/etc/redis-secrets",
    "readOnly": true
   }]
  "volumes": [{
   "name": "foo",
   "secret": {
    "secretName": "mysecret"
   }
  }]
 }
}
```

But Even The Good Solutions….

- By default, Kubernetes stores your credentials in plain text on the server in **etcd**
  - **etcd** is a distributed key-vaue store
  - Stores and replicates cluster state
- However, this is a lot better than nothing
- Proposal to encrypt secrets at-rest
  - https://github.com/kubernetes/community/pull/454

OpenShift

Managing Secrets on OpenShift – Vault Integration

MAY 9, 2017 BY RAFFAELE SPAZZOLI

Tweet G+1 Like Share 3

Introduction

Credentials are environment dependent configurations that need to be kept secret and should be read only by subjects with a need-to-know.

A previous blog post talked about how to manage environment dependent configuration when building delivery pipelines – these approaches are not adequate for secrets because they don't guarantee confidentiality.

OpenShift offers secrets as a way to inject credentials. Secrets behave as encoded-64 configmaps. From a security perspective, they have the following limitations (as of release 3.5):

1. They are not encrypted at rest.
2. By default, cluster admins can see all the secrets of all the tenants.
3. When in use (i.e. mounted as tempfs in the node that runs the pod that is using them), they can be seen by a node administrator.
4. When in use, they can be seen by anyone who has the ability to remote shell into the container.

Summary

- Increased complexity, with security opportunities
- Things are more likely to spin out of control vs. monolithic apps if you don’t get a handle on them early
- It’s important to make security fit into microservices, not the other way around

Thanks! Keep In Touch

- Email – jack@nvisium.com
- Twitter – https://twitter.com/jack_mannino
- Linkedin - https://www.linkedin.com/in/jackmannino/

Thank You to Our Sponsors

CME Group

Hewlett Packard Enterprise

SYNOPSYS®

ARXAN

BLACKDUCK

CHECKMARX

CITRIX®

FLEXERA SOFTWARE

kiuwan

netsparker®

POSITIVE TECHNOLOGIES

RAPID7

Sonatype

Twistlock

Invest Northern Ireland

INTUITION TechChannels

jscrambler

PROMON

WhiteHat SECURITY

MCSGROUP
INTELLIGENT RECRUITMENT SOLUTIONS

edgescan

Invest Northern Ireland

Allstate
You're in good hands.