AWS

SUMMIT

Automating Security in Cloud Workloads with DevSecOps

Chris McCurdy, Global Solutions Architect
(Healthcare and Life Sciences)

28 June 2017

amazon web services

What to expect from the session

Why security automation

Who, security team in a DevSecOps world

Where do you want security automation

When – Pre, post and everything in between

What can you do, practical examples

How – Tools and partners

Terminology Disclaimer

```
import re
re.search('([Dd]ev[Ss]ec|[Ss]ec[Dd]ev|[Rr]ugged\s[Dd]ev)[Oo]ps')
```

=

Security Automation

Terminology Disclaimer

```
import re
re.search('([Dd]ev[Ss]ec|[Ss]ec[Dd]ev|[Rr]ugged\s[Dd]ev)[Oo]ps')
```

=

Security Automation

At Scale

Why?

Why - Goals of DevSecOps

Pace of Innovation…meet Pace of Security Automation

Scalable infrastructure needs scalable security

Risk/rating based actions

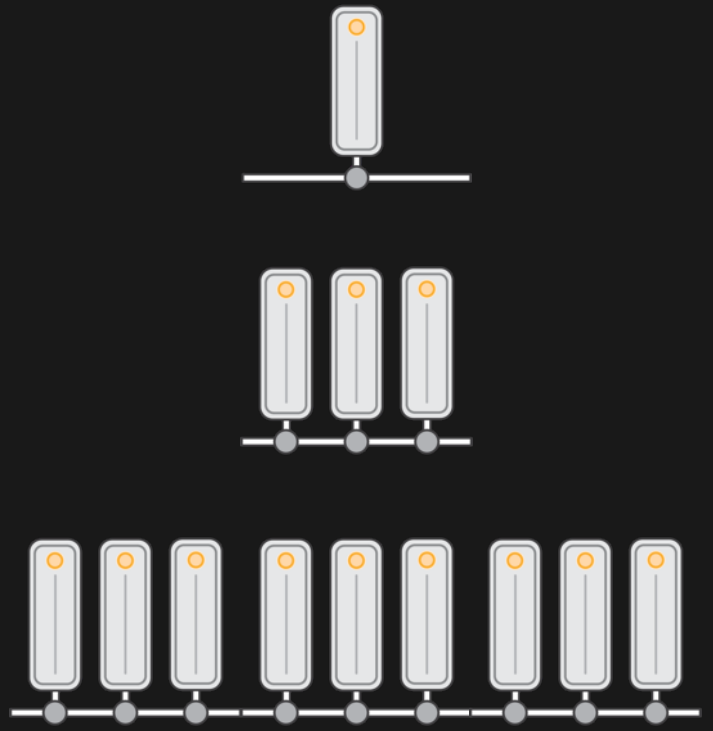
Automatic Incident Response Remediation

Why security automation

Reduce risk of human error

- Automation is effective
- Automation is reliable
- Automation is scalable

Don’t worry…we still need humans



Who?

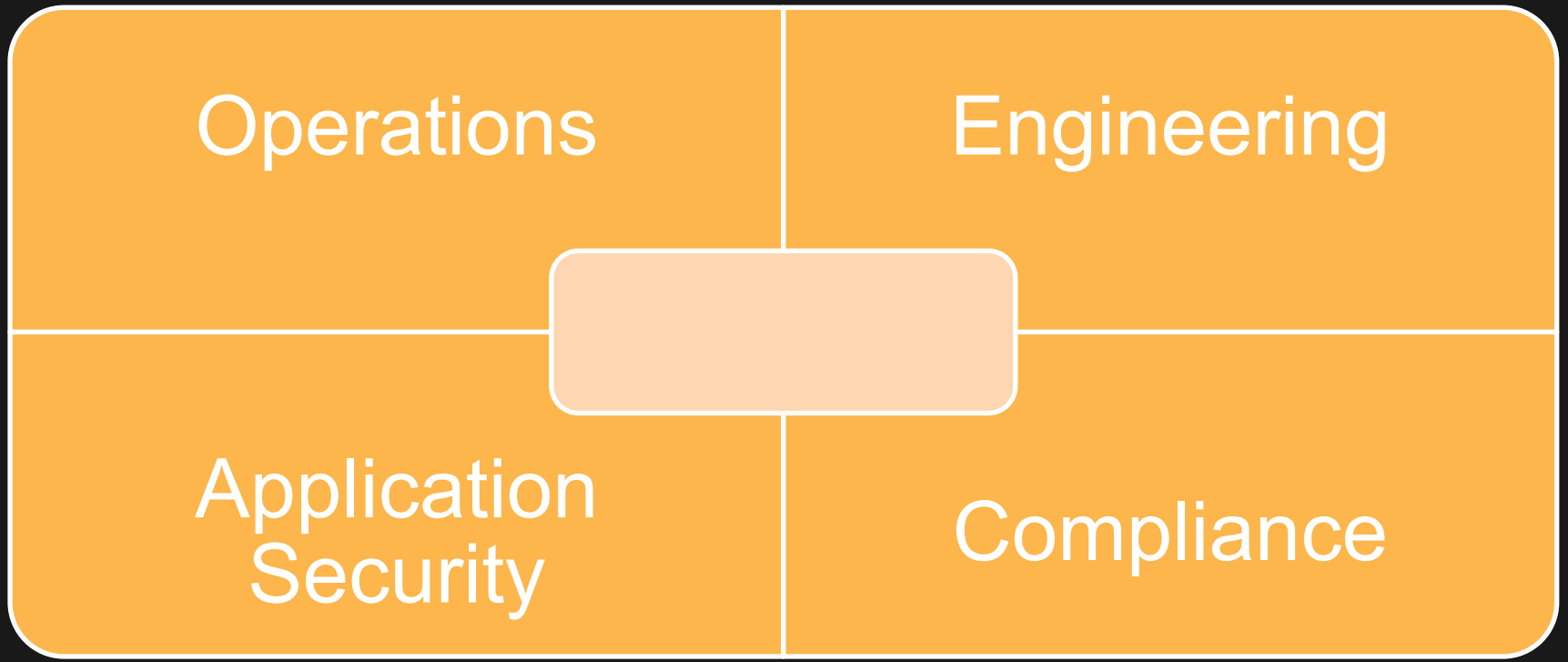
Purpose

Security is a service team, not a blocker

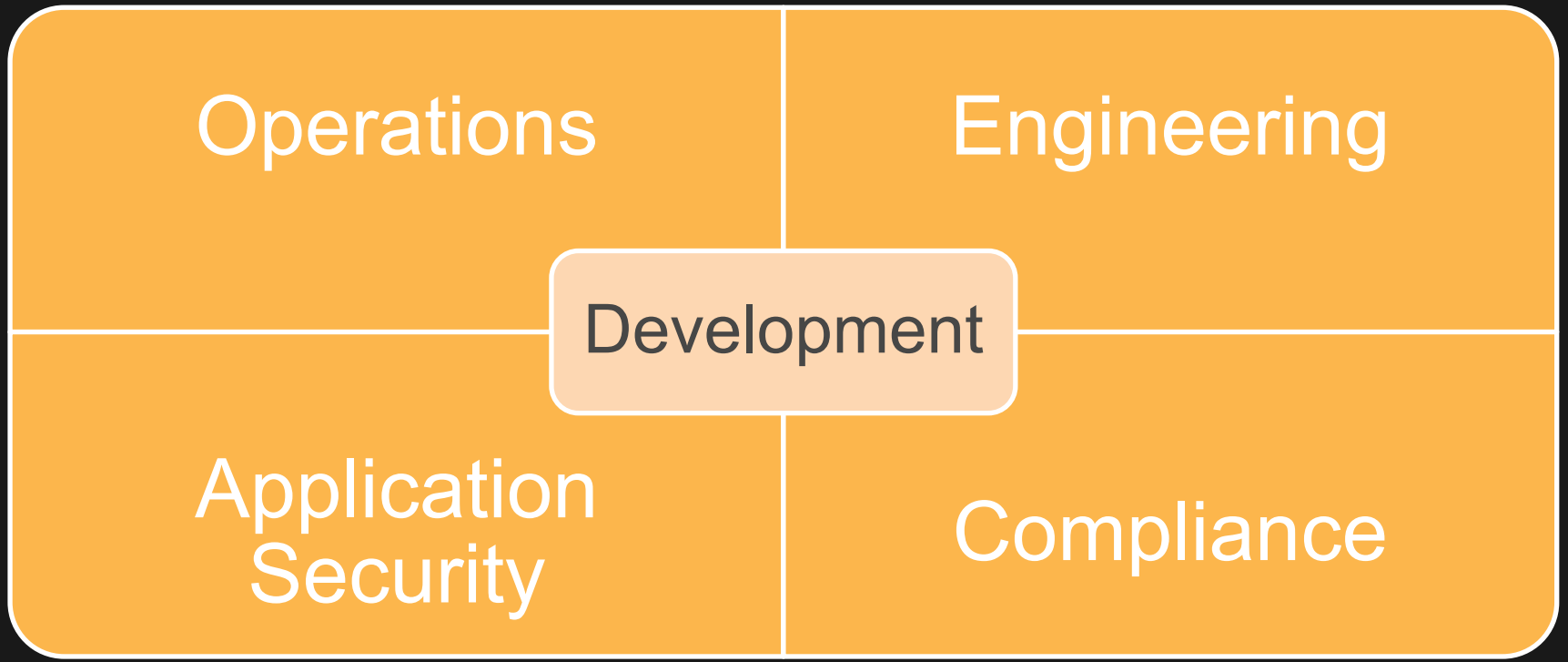
Security is everyone's job

Allow flexibility and freedom
but control the flow and result.

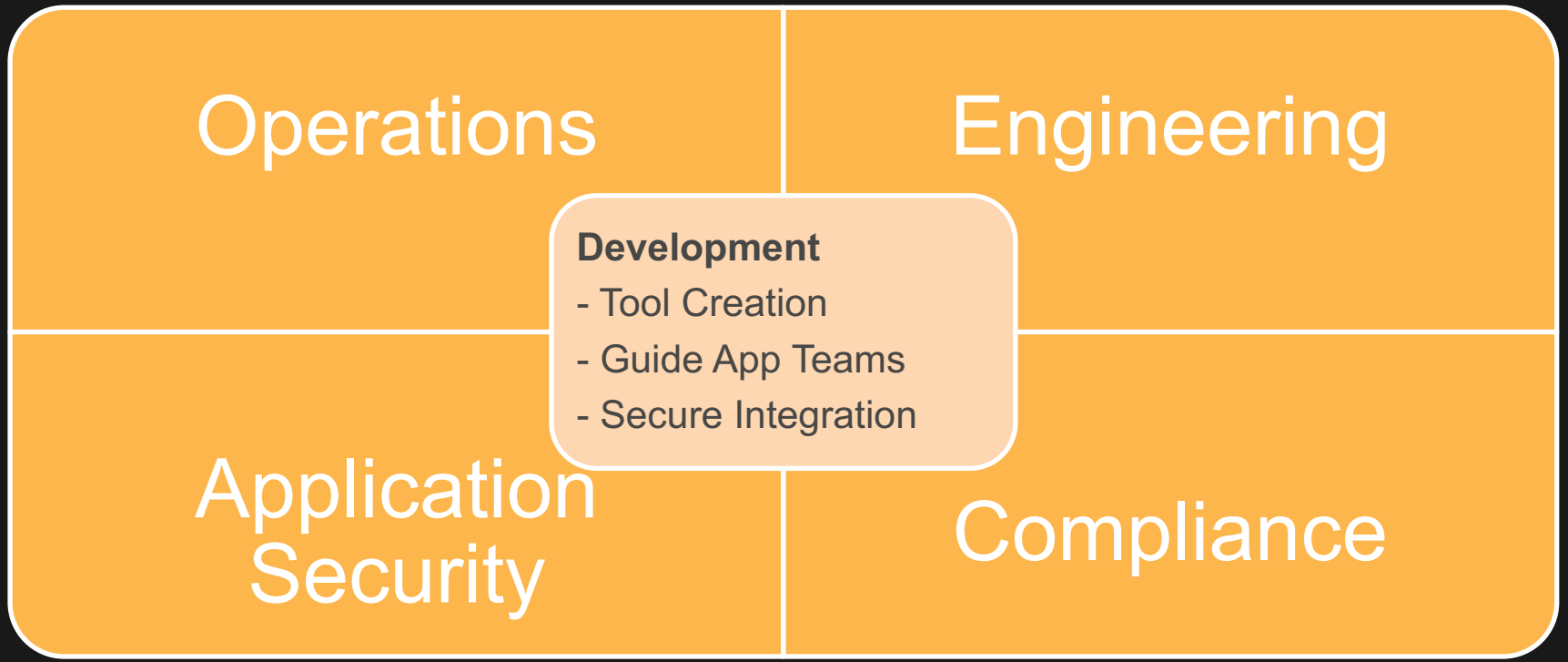
Meet the new security team



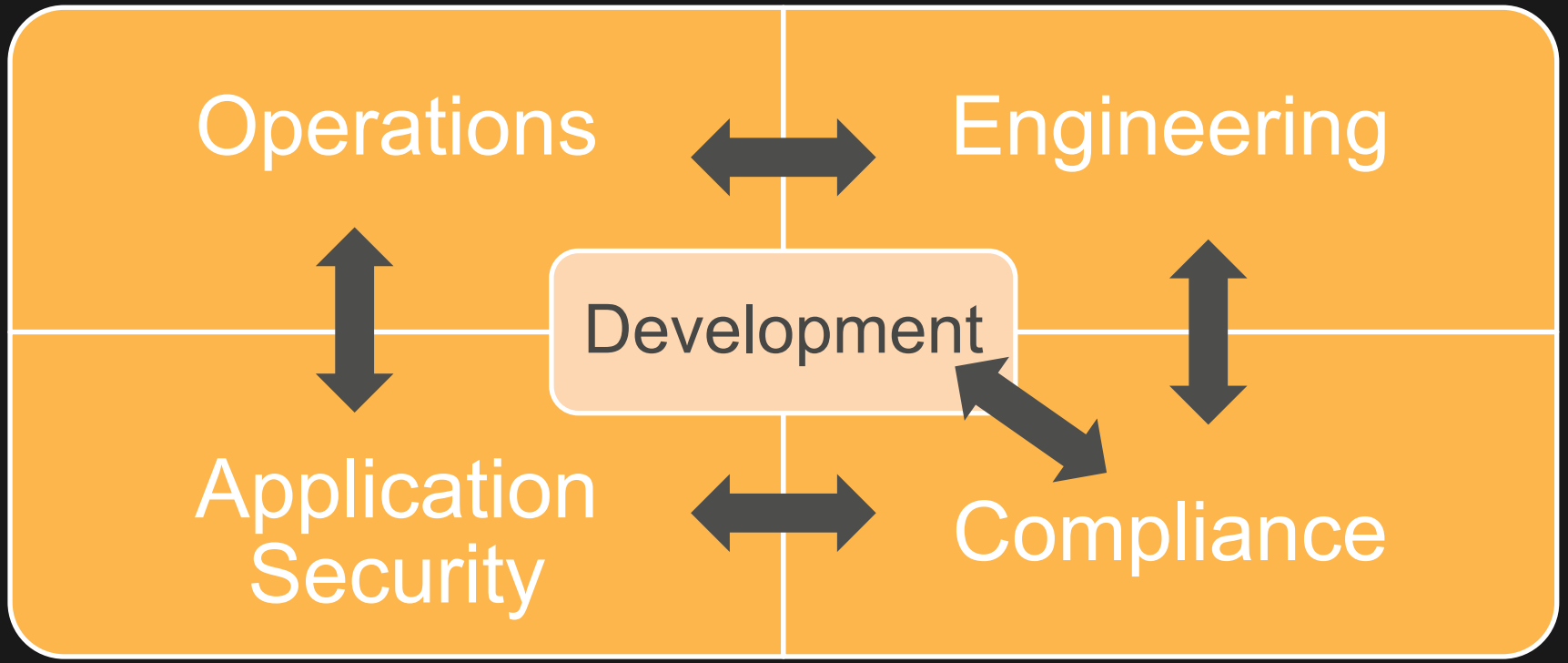
Meet the new security team



Meet the new security team



All roles overlap



Where

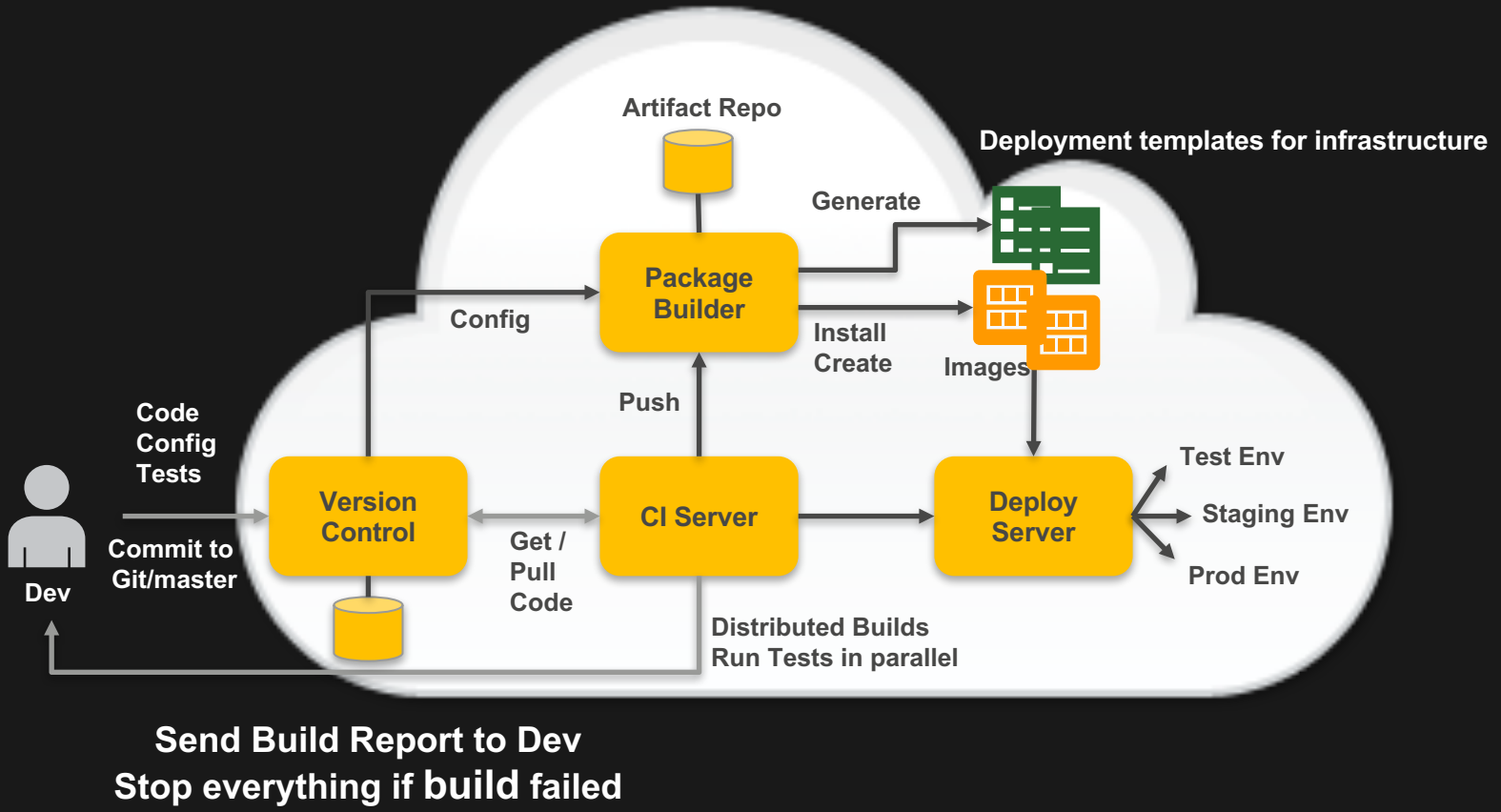
3(+) places

Continuous Integration / Continuous Deployment

1. Security of the CI/CD Pipeline
   - Access roles
   - Hardening build servers/nodes

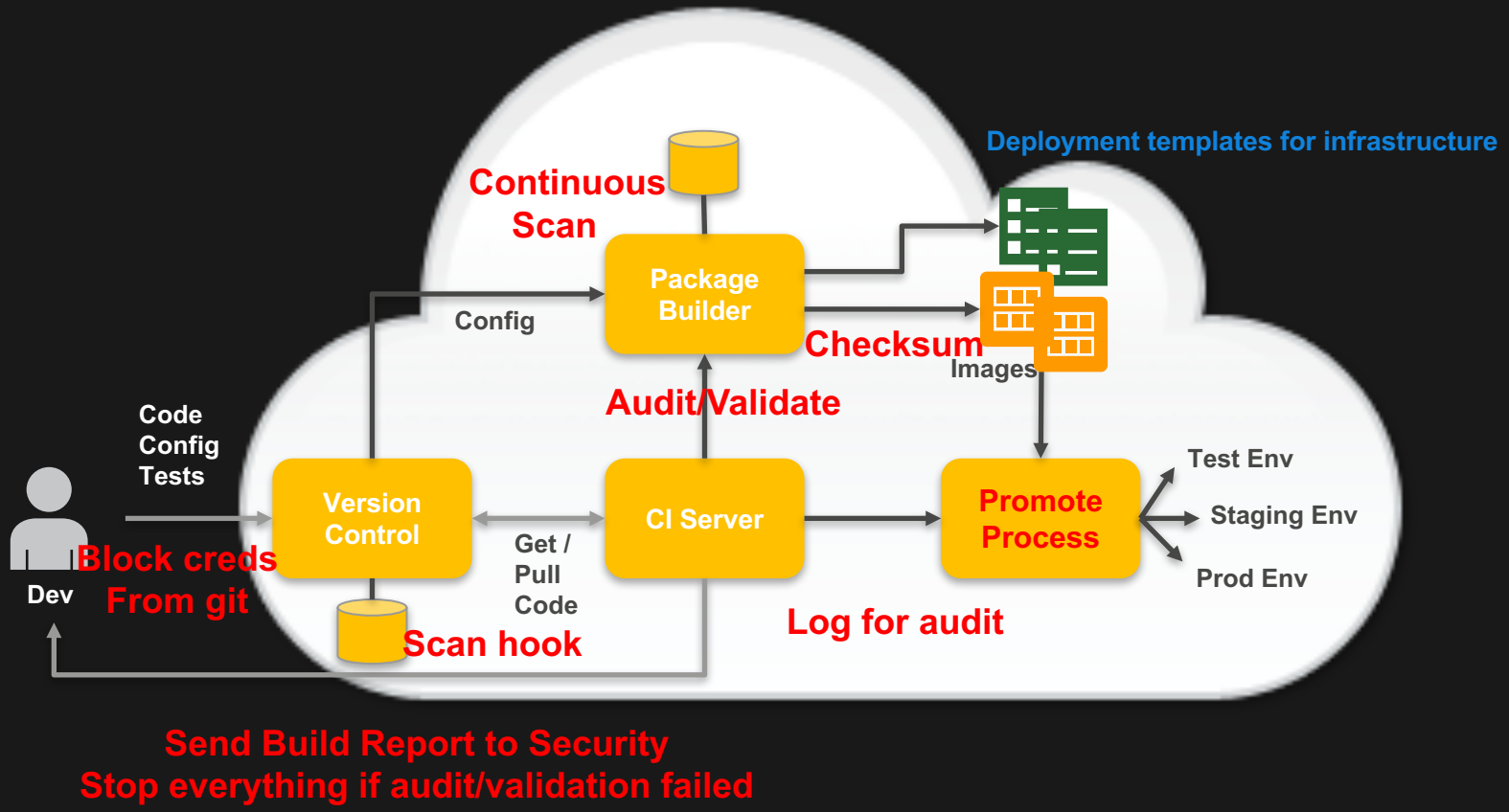
2. Security in the CI/CD Pipeline
   - Artifact validation
   - Static code analysis

CI/CD for DevOps



Send Build Report to Dev
Stop everything if build failed

CI/CD for DevSecOps



What about my other stuff?

3. Cloud scale Security

aka all the other stuff people are really talking about

Infrastructure as code

- Base requirement!
- Split ownership
- Pre-deploy validation

Elastic security automation

- API driven
- Autoscaling groups – hooks
- Execution layer scales with targets

Run time security

- Tag based targeting
- Rip-n-replace

Immutable infrastructure

- Validation and enforcement
- Integrate with managed services

...

When

When

Easy

All the time!

When – Control and Validate

Pre-event - When possible

- Store infrastructure in code repository
  - Validate each push (git hooks)
  - Use managed microservices as execution engine
  - Scan cloud infrastructure templates for unwanted/risk valued configurations
  - Validate Container definitions
- Validate system code early on
  - Find unwanted libraries etc.
- Force infrastructure changes through templates
- Block if needed/unsure

When – Control and Validate

Post-event - Always

- Follow-up on sensitive API's
  - IAM, Security Groups/Firewall, Encryption keys, Logging, etc.
  - Alert/Inform
- Use source of truth
  - Locked to execution function (Read Only)
- Validate source
  - Human or Machine/CICD
- Decide on remediation

When – Control and Validate

Triggers – Event based:

- Per change
  - API based
  - Event logs
- Per day
- Per framework
  - Overall infrastructure, components and resources
  - One component multiple frameworks

What

Give me some examples

Give me some examples

Security validation in a elastic infrastructure

- Implement -> Validate -> Decide
- Terminate upon failure

Automatic Incident Response Remediation

- Autoheal Cloudtrail logging
- Disable offenders

Integrate host-based action with cloud-based control

- Immutable infrastructure – Auto isolate instances

Example – Auto isolation

Modify

- /etc/pam.d/sshd

Execute script upon logon

- *session optional pam_exec.so /path/trigger.sh*

Trigger AWS event as marker using IAM roles for EC2

```
#!/bin/bash
INSTANCE_ID=$(wget -q -O - http://169.254.169.254/latest/meta-data/instance-id)
REGION=$(wget -q -O - http://169.254.169.254/latest/meta-data/placement/availability-zone|sed 's/.\\{1\\}$//')DATE=$(date)
aws ec2 --region $REGION create-tags --resources $INSTANCE_ID --tags \"Key=Tainted,Value=$DATE\"
```

Execute Lambda function using CloudWatch Events on marker detection

- Remove from load balancer/scaling groups (will auto-heal)
- Block in/outgoing traffic using security groups and ACL

Example – Auto isolation

Don’t forget safeguards!

- How many instances can I isolate before failure
- If isolated > x:
  wake_human()
- Remember, x could be 0

Example logging

Detect
- Cloud logging disabled

Priority
- Enable logging

Forensics
- Have this happened before

Countermeasures
- If num_disabled > x: # x could be zero based on type and user
  disable_user()
  - Safeguard: Should I temporary disable user? Who is the user?

Alert!

How

Partners - Security subcategories

NETWORK SECURITY	SECURITY INTELLIGENCE	IDENTITY & ACCESS MANAGEMENT	SECURITY ORCHESTRATION	SERVER / ENDPOINT	DATA SECURITY	APPLICATION SECURITY
Check Point SOFTWARE TECHNOLOGIES LTD.	splunk>	okta	evident.io	TREND MICRO	gemalto security to be free	IMPERVA
Provides customers with uncompromised protection against all types of threats, reduces security complexity and lowers total cost of ownership.	Easy, fast and secure way to search, analyze and visualize massive data streams	Okta is an integrated identity and mobility management service	Cloud-native infrastructure security solution providing full coverage of all AWS accounts, services and regions	Get hourly proactive protection for your AWS workloads with Trend Micro Deep Security	Protection of data, digital identities, payments, and transactions from the edge to the core	Imperva SecureSphere WAF for AWS extends all of the security and management capabilities of the world's most-trusted web application firewall to Amazon Web Services environments
paloalto NETWORKS	sumologic	onelogin	Dome9 SECURITY		Vormetric	Barracuda
Quickly create a hybrid architecture that extends your existing data center into AWS via encrypted tunnels	With Sumo Logic, you can collect, compress, and securely transfer all of your log data regardless of volume, type, or location	OneLogin, the innovator in Identity and Access Management-as-a-Service (IDaaS)	Dome9 automates AWS security groups and adds an extra layer of protection against hackers		Proactive security from a single agent designed for AWS	Many AWS-hosted applications choose Barracuda, an AWS Preferred Security Competency Partner, due to its continuous monitoring and policy tuning by world-class security experts
Other popular solutions: Check Point, Fortinet, Alert Logic	**Other popular solutions:** Fortinet	**Other popular solutions:** Bitium, ClearLogin, Ping Identity	**Other popular solutions:** Tenable, Qualys	**Other popular solutions:** Symantec, Unisys	**Other popular solutions:** HyTrust, CTERA	**Other popular solutions:** Fortinet

SaaS Subscriptions

Dozens of SaaS applications addressing multiple use cases

Two ways to subscribe to SaaS products

PAY-AS-YOU-GO SUBSCRIPTIONS (MARKETPLACE METERING SERVICE)

- Buyers can easily find and subscribe to SaaS products in Marketplace. As they use the software, Seller sends metering records summarizing usage to AWS.
- AWS adds to the Buyer's monthly bill, based on metered data sent by Seller.
- Launched November, 2016

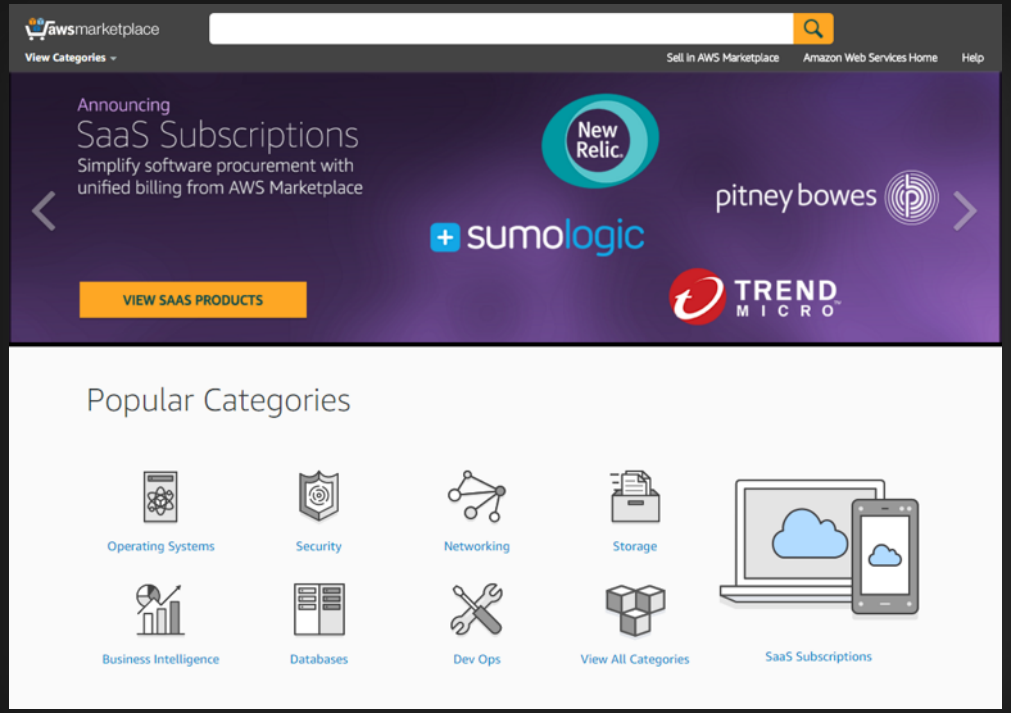
PRE-PAID SUBSCRIPTIONS (CONTRACTS)

- Buyers can purchase monthly, yearly, or multi-year subscriptions that automatically renew through a shopping-cart experience. User provisioning and account setup continues within the seller's application.
- Payment occurs up front. Buyers can increase the size of contracts at any time, adding to their existing renewal date at the pro-rated cost.
- Launching in April, 2017

AWS Marketplace

Discover, Procure, Deploy, and Manage Software In the Cloud

- 3,600+ software listings
- 51 SaaS paid SaaS Products
- Over 1,100 participating ISVs
- Deployed in 14 AWS Regions
- 100,000+ active customers
- Over 300M of deployed EC2 instances per month
    - That's 400K per hour
- Curated Products
- Integrated to AWS Billing



Putting it all together

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

Storage
- S3
- EFS
- Glacier
- Storage Gateway

Database
- RDS
- DynamoDB
- ElastiCache
- Redshift

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

Migration
- Application Discovery Service
- DMS
- Server Migration
- Snowball

Developer Tools
- CodeStar
- CodeCommit
- CodeBuild
- CodeDeploy
- CodePipeline
- X-Ray

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

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

Contact Center
- Amazon Connect

Game Development
- Amazon GameLift

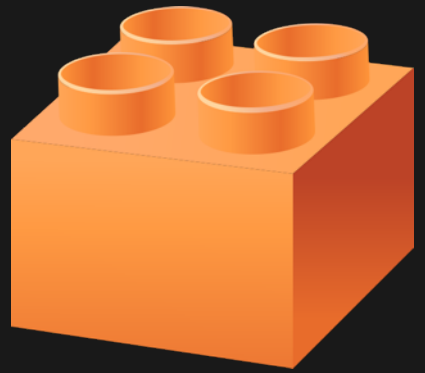
Mobile Services
- Mobile Hub
- Cognito
- Device Farm
- Mobile Analytics
- Pinpoint

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

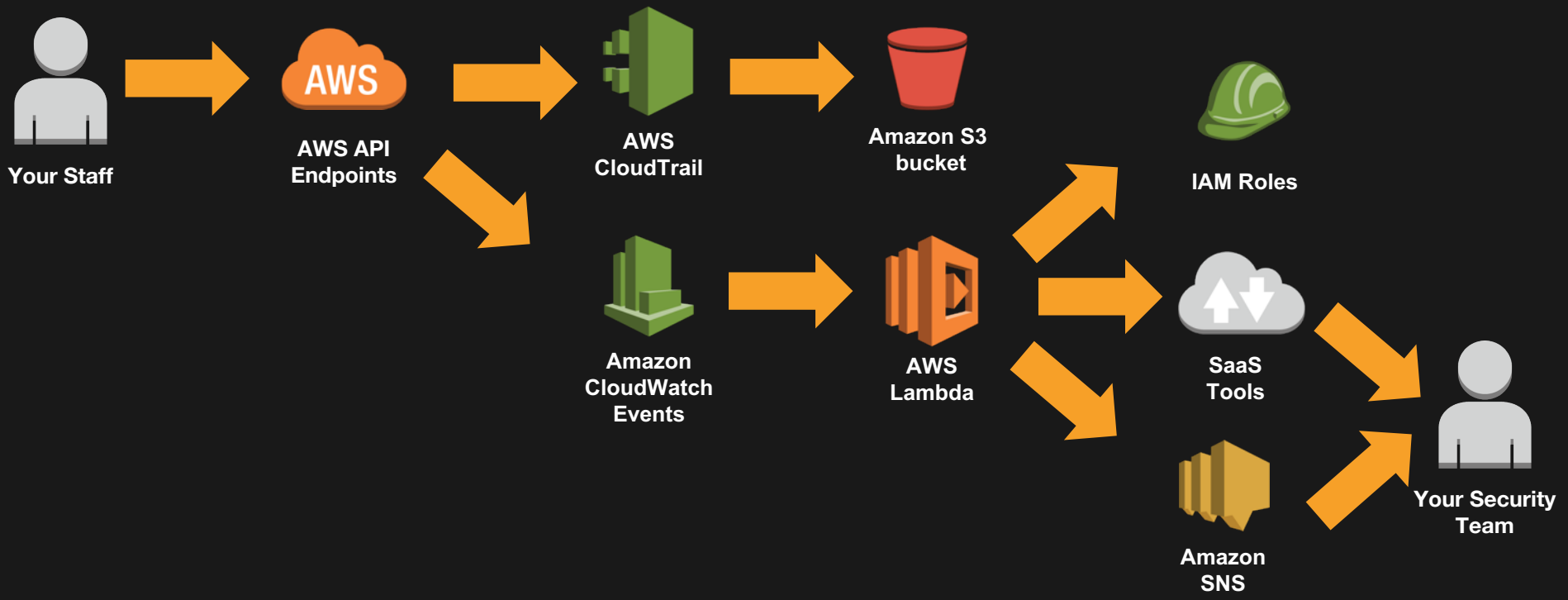
Messaging
- Simple Queue Service
- Simple Notification Service
- SES

Business Productivity
- WorkDocs
- WorkMail
- Amazon Chime

Desktop & App Streaming
- WorkSpaces
- AppStream 2.0



Putting it all together



AWS Tooling

Execution

- Lambda

Tracking

- AWS Config Rules
- Amazon CloudWatch Events
- AWS Step Functions
- AWS CloudTrail
- AWS Inspector

Track/Log

- Amazon CloudWatch Logs
- Amazon DynamoDB

Alert

- SNS

Third party Open Source

Cool…so I just fix things??

Well…yes...but...

Risks

Failure is always an option, now at script speed

We forgot to tell you…

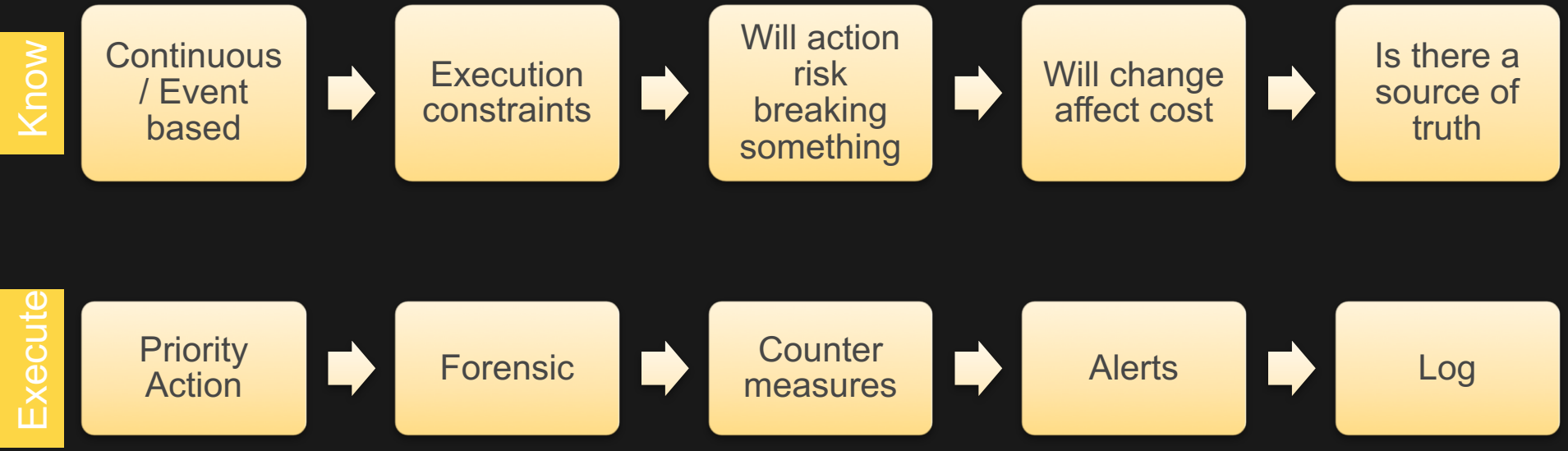
No proper alerting, logging or follow-up on automated events

You got scripts…they got scripts

How do you minimize risk of failed remediation functions?

Implement remediation framework

The anatomy of remediation



What else can I do

Benchmarking infrastructure

Map your infrastructure against control frameworks

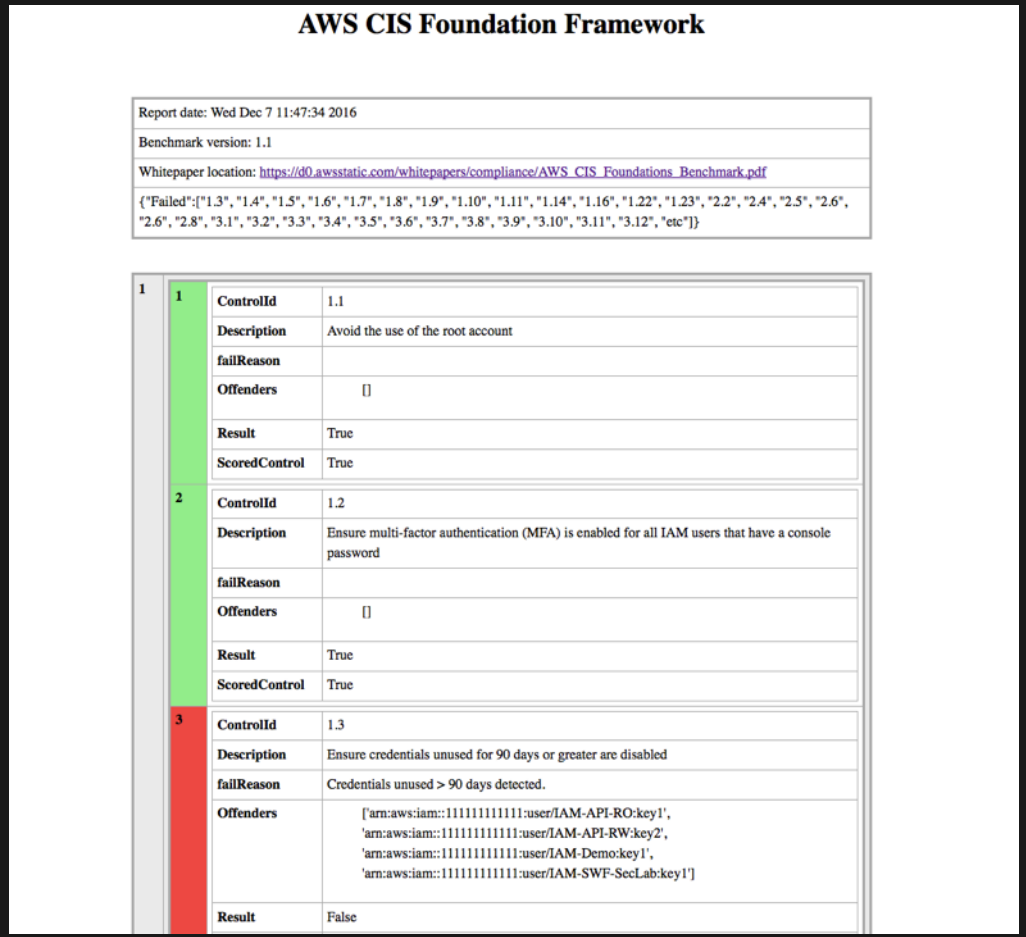
Single run for single account health check

AWS Config / Config Rules for compliance tracking

Example: OSS validation for CIS AWS Foundation Framework

- https://github.com/awslabs/aws-security-benchmark

Report this way…



AWS CIS Foundation Framework

Report date: Wed Dec 7 11:47:34 2016
Benchmark version: 1.1
Whitepaper location: https://d0.awsstatic.com/whitepapers/compliance/AWS_CIS_Foundations_Benchmark.pdf
{"Failed":["1.3", "1.4", "1.5", "1.6", "1.7", "1.8", "1.9", "1.10", "1.11", "1.14", "1.16", "1.22", "1.23", "2.2", "2.4", "2.5", "2.6", "2.6", "2.8", "3.1", "3.2", "3.3", "3.4", "3.5", "3.6", "3.7", "3.8", "3.9", "3.10", "3.11", "3.12", "etc"]}

1

1

ControlId	1.1
Description	Avoid the use of the root account
failReason	
Offenders	[]
Result	True
ScoredControl	True

2

ControlId	1.2
Description	Ensure multi-factor authentication (MFA) is enabled for all IAM users that have a console password
failReason	
Offenders	[]
Result	True
ScoredControl	True

3

ControlId	1.3
Description	Ensure credentials unused for 90 days or greater are disabled
failReason	Credentials unused > 90 days detected.
Offenders	['arn:aws:iam::111111111111:user/IAM-API-RO:key1', 'arn:aws:iam::111111111111:user/IAM-API-RW:key2', 'arn:aws:iam::111111111111:user/IAM-Demo:key1', 'arn:aws:iam::111111111111:user/IAM-SWF-SecLab:key1']
Result	False

Or this…


```
{
    "1": {
        "1": {
            "ControlId": "1.1",
            "Description": "Avoid the use of the root account",
            "Offenders": [],
            "Result": true,
            "ScoredControl": true,
            "failReason": ""
        },
        "2": {
            "ControlId": "1.2",
            "Description": "Ensure multi-factor authentication (MFA) is enabled for all IAM users that have a console password",
            "Offenders": [],
            "Result": true,
            "ScoredControl": true,
            "failReason": ""
        },
        "3": {
            "ControlId": "1.3",
            "Description": "Ensure credentials unused for 90 days or greater are disabled"
```

Or maybe just this

{"Failed":["1.3", "1.4", "1.5", "1.6", "1.7", "1.8", "1.9", "1.10", "1.11", "1.14", "1.16", "1.22", "1.23", "2.2", "2.4", "2.5", "2.6", "2.6", "2.8", "3.1", "3.2", "3.3", "3.4", "3.5", "3.6", "3.7", "3.8", "3.9", "3.10", "3.11", "3.12", "etc"]}

Or maybe just this

{"Failed":["1.3", "1.4", "1.5", "1.6", "1.7", "1.8", "1.9", "1.10", "1.11", "1.14", "1.16", "1.22", "1.23", "2.2", "2.4", "2.5", "2.6", "2.6", "2.8", "3.1", "3.2", "3.3", "3.4", "3.5", "3.6", "3.7", "3.8", "3.9", "3.10", "3.11", "3.12", "etc"]}

Control output based on consumer of data and post processing of result

At the end of the rainbow…

What are we trying to accomplish?

Goals

Minimize relying on humans for active security events

- Automation doesn’t sleep, eat or need coffee in the morning

Prevent bad configurations before they are implemented

Autocorrect/remediate violations where possible

Daily/instant benchmark validation of infrastructure

- Validate against industry frameworks
- Extend to remediation

Your next step

Look through your infrastructure security runbook

- What can you automate?
- How can you validate?

OSS Code to learn from

git-secrets - Prevents you from committing passwords and other sensitive information to a git repository.

aws-security-benchmark - Benchmark scripts mapped against trusted security frameworks.

aws-config-rules - [Node, Python, Java] Repository of sample Custom Rules for AWS Config

Netflix/security_monkey - Monitors policy changes and alerts on insecure configurations in an AWS account.

Netflix/edda - Edda is a Service to track changes in your cloud deployments.

ThreatResponse - Open Source Security Suite for hardening and responding in AWS.

CloudSploit – Capturing things like open security groups, misconfigured VPCs and more.

Stelligent/Cfn_nag – Looks for patterns in CloudFormation templates that may indicate insecure infrastructure.

Capitalone/cloud-custodian - Rules engine for AWS fleet management.

AWS

SUMMIT

MIRACL - Securing Workloads in the Cloud

Kevin Brannigan

Director - DevOps & Partner Engineering, MIRACL

amazon web services

MIRACL - Introduction

Who we are, what we do: Short Introduction

How MIRACL provides security: Distribution of Trust

How we leverage AWS for over 100k authentications/sec

How we secure our infrastructure: Tools and Glue

How we orchestrate containers: Containers Rule

How we secure IOT: Channel Security (TLS) without X509 Certs

Live Demo!

MIRACL Secures The People, Apps, And Things Needed To Run A Digital Business

WEB AND MOBILE APPS
Remove passwords threats completely

CLOUD SERVICES
Establish integrity in cloud security

INTERNET OF THINGS
Instant trust between “things”

FINTECH / BLOCKCHAIN
Deliver transaction speed and security

ALGORITHMIC BUSINESS / AI
Ensure data integrity

Securing Internal & External Systems

CHANNEL SECURITY

A single platform for Internal and External Authentication and Security.

SINGLE SIGN ON

AUTHENTICATION

Channel Security (TLS) at scale without X509 Certs, APP to APP, Machine to Machine security.

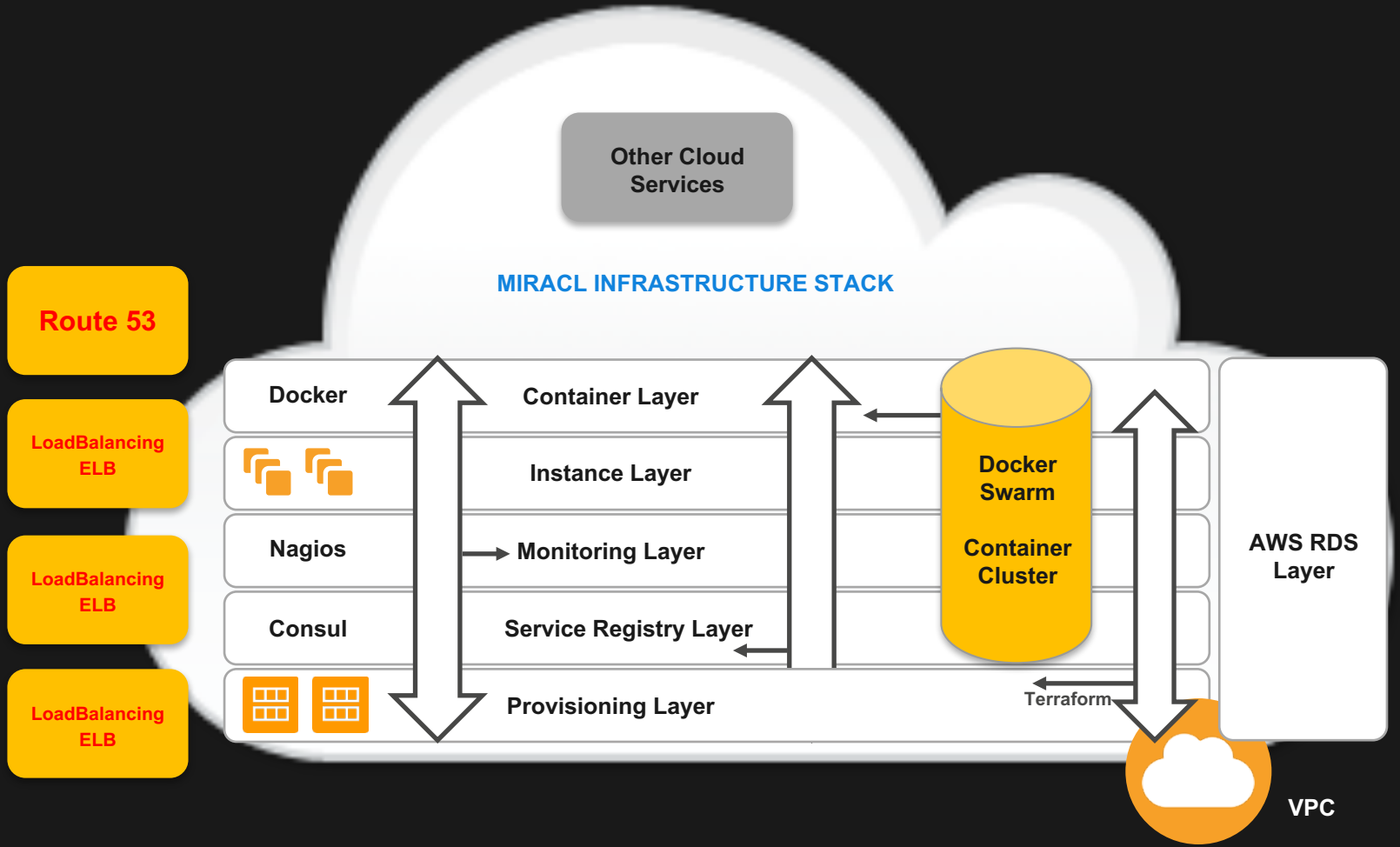
SECURE SHELL

VPN SERVICES

No SSH Keys, One Time Password for VPN & SSH, Secure AWS and Other Console Access.

MIRACL - How we leverage AWS for > 100k auths/sec

- Terraform to provision cloud infrastructure
- Multi-cloud, fast, repeatable, reliable, infrastructure as code
- Build on demand: VPCs, subnets, instances, RDS DBs ElastiCache, Route 53, CDN assets, S3 buckets
- Scratch Containers with Go Binaries <2mb size, <1024mem
- Minimal Containers inc. Casper, Consul
- Docker Swarm
- NATS message bus (cluster mode, scalable)
- Consul for key-value, service discovery
- Vault & AWS KMS for key material
- Microservices platform - horizontal scalability

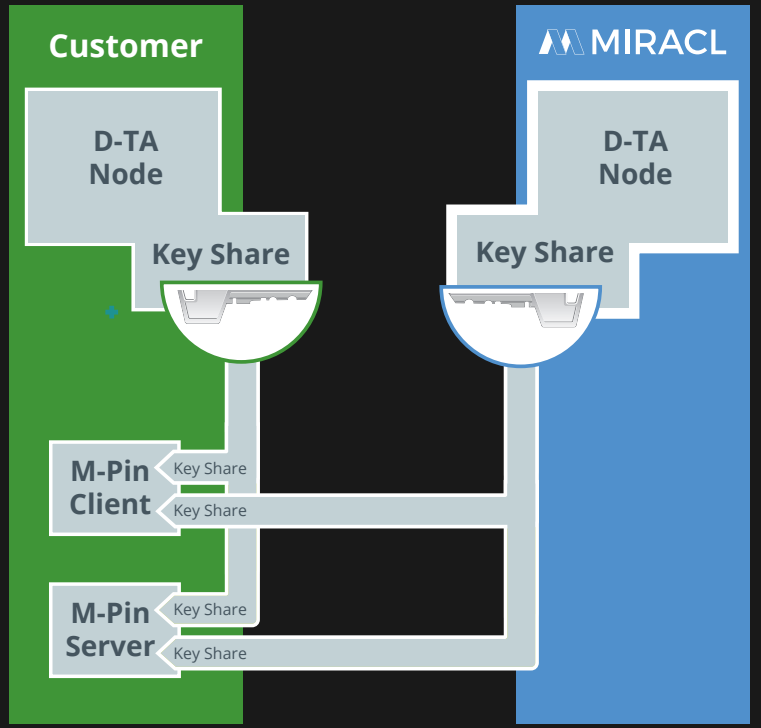

Other Cloud Services
MIRACL INFRASTRUCTURE STACK
Route 53
LoadBalancing ELB
LoadBalancing ELB
LoadBalancing ELB
Docker
Container Layer
Instance Layer
Nagios
Monitoring Layer
Consul
Service Registry Layer
Provisioning Layer
Terraform
Docker Swarm
Container Cluster
AWS RDS Layer
VPC

MIRACL - How we secure our infrastructure

- Nothing sensitive in public repos
- Nothing sensitive in AMIs
- All sensitive info in Vault
- AWS, JIRA, Tools are secured with our SSO
- SSH and VPNs secured with our SSO - OTP
- No password database!
- DTA keys self-rotating with ‘whack a mole’
- MTLS -- All traffic and communications
- Multifactor Authentication on Everything

Everything is Distributed

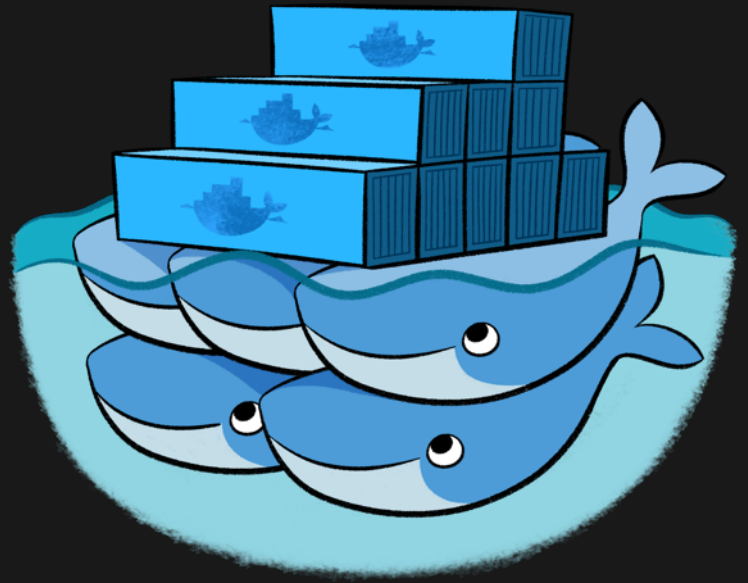
- MIRACL DTA's provide pieces of keys
- Distributed GIT Repos
  (Separate Regions / Clouds)
- Separate Systems for Open Source and Corporate
- Vault AAS - Consul Backend across zones
- Separate Key Material and methods, KMS and Vault
- Separate Roles and Access OPS vs. Dev



MIRACL Trust® Distributed Trust Authorities (D-TAs).

MIRACL - How we orchestrate containers

- Why Docker Swarm?
- Out of the box security
- Mesh networking
- Portability to Docker Compose (single user versions)
- Built in health checks
- Restart and update policies
- Consul for service registration



MIRACL - Live Demo

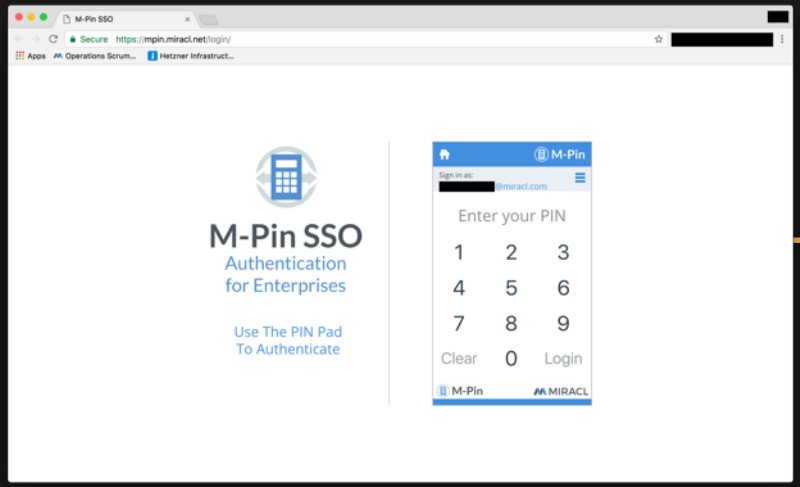
A live demo of how we build on demand:

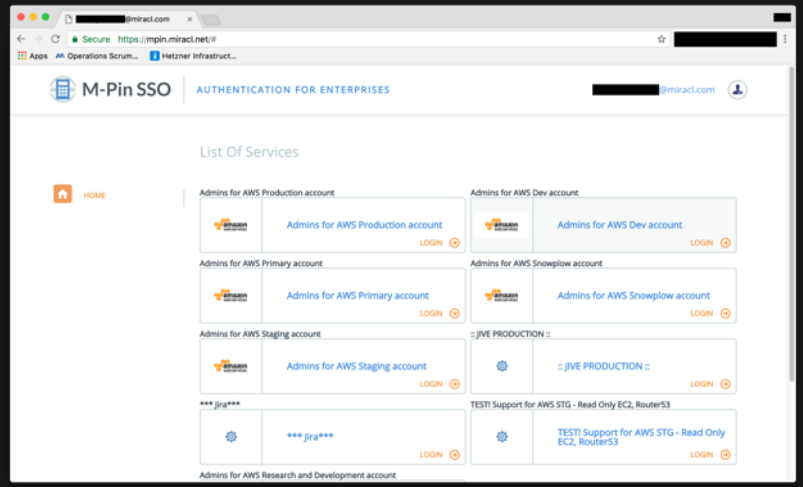
- AWS infrastructure deployed by Terraform
- Container based
- Docker Swarm, etc.

MIRACL - How we secure IoT

- M-PIN Full for key distribution, i.e. creating a PSK
- Secure channel using TLS-PSK without certificates
- MQTT protocol, broker on our platform
- End-to-end encryption between devices

MIRACL - What we do: SSO





MIRACL - How it works: MFA and DTAs

- No passwords, so no hashes stored, so cannot be stolen or lost
- Multiple Distributed Trust Authorities (DTAs) in different cloud providers – no single point of compromise
- Multi-factor authentication
- Something you have - client key (e.g. mobile device)
- Something you know - PIN
- M-PIN authentication protocol
- Zero-knowledge proof - keys not exchanged

MIRACL - References

MIRACL
https://www.miracl.com

MIRACL Crypto Library
https://libraries.docs.miracl.com/

GitHub
https://github.com/miracl/MIRACL

AMCL White Paper
https://github.com/miracl/milagro-crypto-c/blob/develop/doc/AMCL.pdf

AWS

SUMMIT

amazon web services