



# A DevOps Approach to Security Controls

Kenneth G. Hartman



# About Me

*“I help my clients earn and maintain the trust of their customers”*

## Kenneth G. Hartman

- BS Electrical Engineering, Michigan Technological University
- MS Information Security Engineering, SANS Technology Institute
- Multiple Security Certifications: CISSP, GIAC Security Expert, etc.
- SANS Instructor – SEC545 Cloud Security Architecture & Operations

[www.kennethghartman.com](http://www.kennethghartman.com)  
@kennethghartman

***The content and opinions in this presentation are my own and do not necessarily reflect the positions, strategies, or opinions of any current or previous employer.***



# Objectives

- **Advocate the use of DevOps Principles for Security Control Implementation.**
- **Promote the use of Security Automation to achieve Continuous Monitoring of all Security Controls.**
- **Use the organization's CI/CD Toolchain to deploy and maintain the Security Automation.**




# Problem Statement

Security Challenges are growing faster than Security Teams

- The ratios of Developers and Operators per Security Engineer is increasing dramatically.
- The ratio of Virtual Machines per Operator is also increasing rapidly.
- Many organizations are deploying new security “challenges” faster than they can be detected and remediated.

Humans cannot remember the content of (or the rationale for) all security policy and will make expedient “tradeoffs.”

Frequent staffing changes leave control ownership gaps.



*"We cannot solve our problems  
with the same level of thinking  
that created them."*

—Einstein



# What is DevOps?

*DevOps is both a culture and a set of processes and tools that enable development and operation teams to create, release, and manage applications at a high velocity following a Systems Development Life Cycle (SDLC) that is typically automated via Continuous Integration/Continuous Delivery (CI/CD) tooling.*



# More on DevOps...

## Benefits of DevOps

- Speed
- Rapid Delivery
- Reliability
- Scale
- Improved Collaboration
- Security

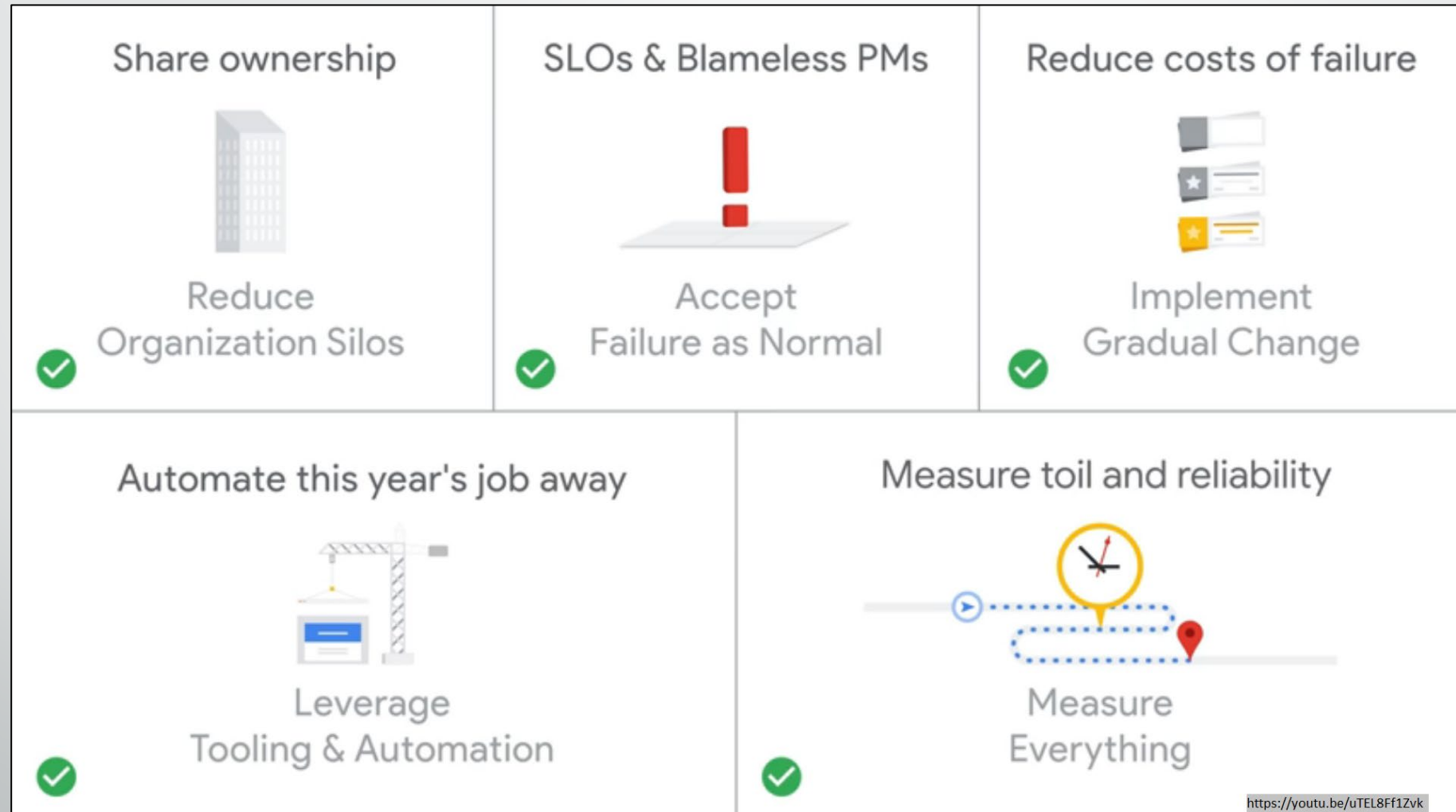
## DevOps Best Practices

- Continuous Integration
- Continuous Delivery
- Microservices
- Infrastructure as Code
- Monitoring and Logging
- Communication and Collaboration

<https://aws.amazon.com/devops/what-is-devops/>



# Google's Take: DevOps vs SRE





# Is it Worth the Time?

[xkcd.com/1205/](http://xkcd.com/1205/)

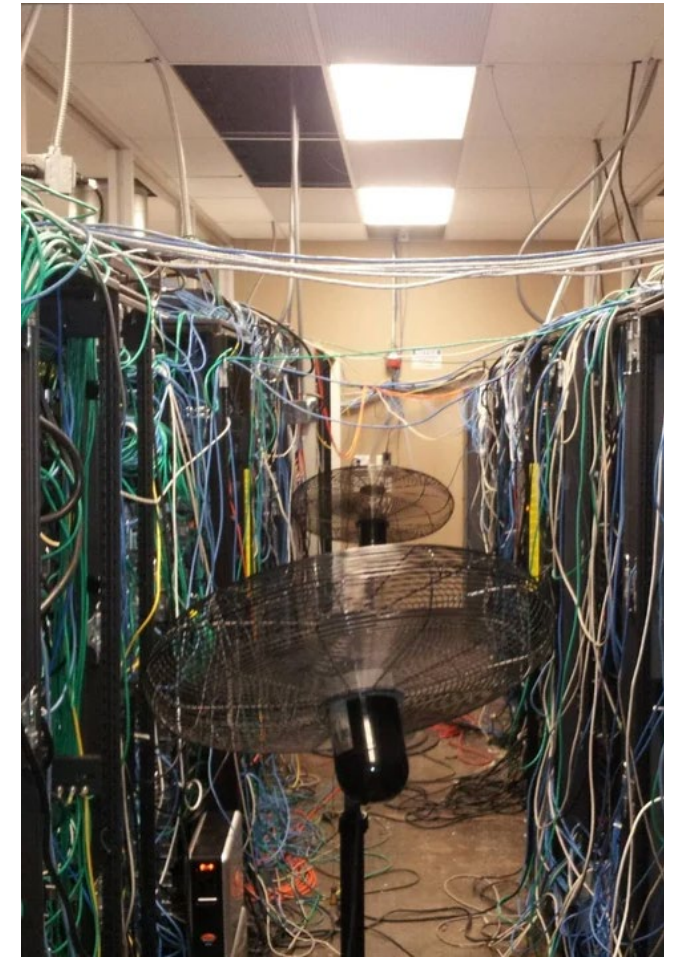
HOW LONG CAN YOU WORK ON MAKING A ROUTINE TASK MORE EFFICIENT BEFORE YOU'RE SPENDING MORE TIME THAN YOU SAVE?  
(ACROSS FIVE YEARS)

		HOW OFTEN YOU DO THE TASK					
		50/DAY	5/DAY	DAILY	WEEKLY	MONTHLY	YEARLY
HOW MUCH TIME YOU SHAVE OFF	1 SECOND	1 DAY	2 HOURS	30 MINUTES	4 MINUTES	1 MINUTE	5 SECONDS
	5 SECONDS	5 DAYS	12 HOURS	2 HOURS	21 MINUTES	5 MINUTES	25 SECONDS
	30 SECONDS	4 WEEKS	3 DAYS	12 HOURS	2 HOURS	30 MINUTES	2 MINUTES
	1 MINUTE	8 WEEKS	6 DAYS	1 DAY	4 HOURS	1 HOUR	5 MINUTES
	5 MINUTES	9 MONTHS	4 WEEKS	6 DAYS	21 HOURS	5 HOURS	25 MINUTES
	30 MINUTES		6 MONTHS	5 WEEKS	5 DAYS	1 DAY	2 HOURS
	1 HOUR		10 MONTHS	2 MONTHS	10 DAYS	2 DAYS	5 HOURS
	6 HOURS				2 MONTHS	2 WEEKS	1 DAY
	1 DAY					8 WEEKS	5 DAYS

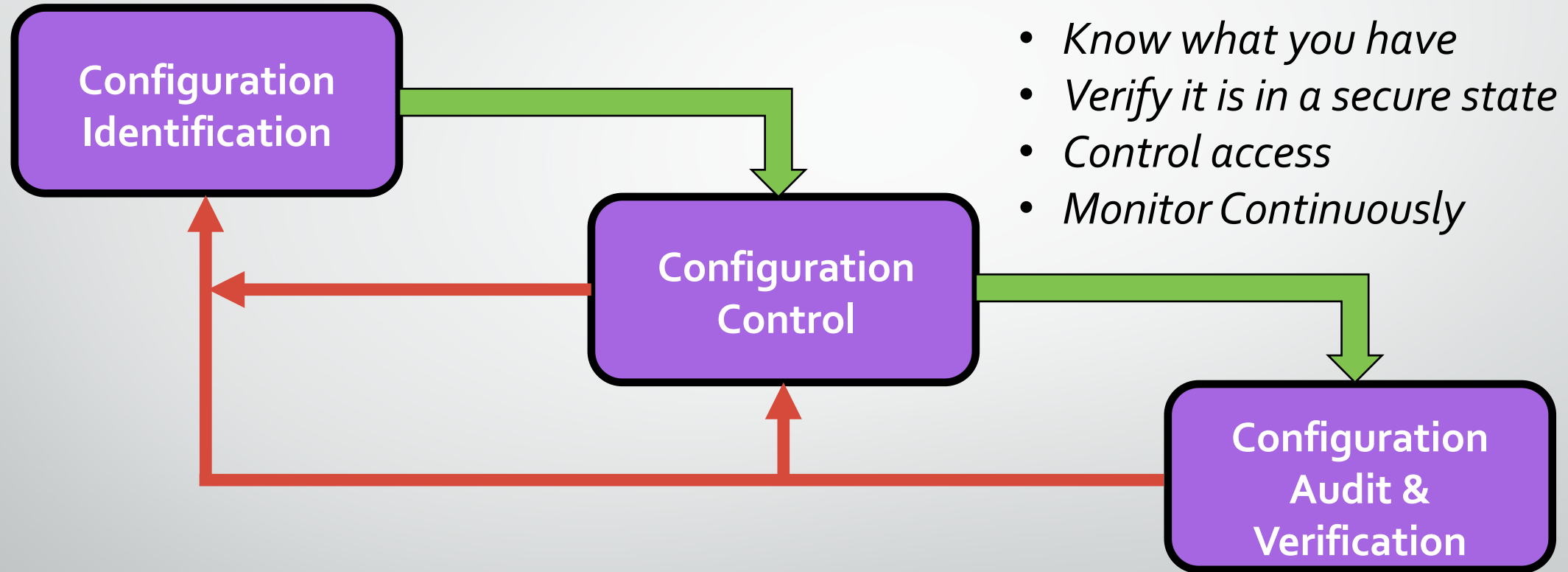
# Best in Class looks Best in Class in Every Way

*Good Hygiene takes  
effort and focus, but  
the payoff is:*

- *reduced waste*
- *operational efficiencies*
- *improved security*
- *sense of ownership.*



# Configuration Management





# Security Compliance Terminology

- **Security Control** – a testable countermeasure designed to mitigate a *specific* risk
- **Defense-in-depth** – Overlapping controls such that if one control fails, others mitigate the risk.
- Complementary controls create a **Security Capability** (such as Access Control or Incident Response) and are an *appreciable* organizational asset.
- **Security assurance** is confidence that an entity meets its security requirements based on specific, tested evidence.



## Types of Security Controls

---

**Preventive** – Prevent the risk from being realized

---

**Detective** – Minimize the risk by taking early action

---

**Corrective** – Restore the systems to normal operation

---

**Forensic** – Determine the root cause of the incident





# Example Policy Requirement #1

## **PCI Requirement 1.2.1**

Restrict inbound and outbound traffic to that which is necessary for the cardholder data environment, and specifically deny all other traffic.

## **Example Company Configuration Standard**

AWS Security Group Egress Rules in production environments shall not be configured for “Allow All to Anywhere.”

How can this policy be enforced programmatically?



# Example Policy Requirement #2

## Example Company Policy

Network environments and virtual instances shall be designed and configured to restrict and monitor traffic between trusted and untrusted connections. These configurations shall be reviewed at least annually and supported by a documented justification for use for all allowed services, protocols, ports, and compensating controls.

### Requirements:

1. Proof of Annual Review → Tag each SG with Last Review Date & Ticket Number
2. Documented Justification → Reference Justification in Rule Description

How can this policy be enforced programmatically?



# Proof of Annual Review

Security Group: sg-009d55977125d9fd5

Description Inbound Outbound **Tags**

Add/Edit Tags

Key	Value	
LastReviewDate	2019-10-25	Show Column
ReviewTicket	ESEC-1234	Show Column
Name	Example	Hide Column

Apply tags programmatically based on ticket closure

# Documented Justification

Security Group: sg-009d55977125d9fd5

Description Inbound Outbound Tags

Edit

Type ⓘ	Protocol ⓘ	Port Range ⓘ	Source ⓘ	Description ⓘ
HTTP	TCP	80	0.0.0.0/0	J0010 - Redirect t...
SSH	TCP	22	10.0.0.0/24	J0012 - Local Admi...
HTTPS	TCP	443	0.0.0.0/0	J0011 - HTTPS Appl...

Programmatically validate that the Description contents is a valid based on a list of pre-defined justifications



# Example Policy Requirement #3

## PCI Requirement 6.5

Train developers at least annually in up-to-date secure coding techniques, including how to avoid common coding vulnerabilities.

## Example Company Policy

Software developers and all other relevant personnel involved in the development of software for [COMPANY] are required to undergo annual training in secure coding techniques for the software platforms(s) with which they work.

How can this policy be enforced programmatically?



<> Code

! Issues 0

🔗 Pull requests 0

📁 Projects 0

📖 Wiki

🛡 Security

📊 Insights

⚙ Settings

Options

Collaborators

Webhooks

Notifications

Integrations & services

Deploy keys

Autolink references

Moderation

Interaction limits

Webhooks / Add webhook

We'll send a `POST` request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, `x-www-form-urlencoded`, etc). More information can be found in [our developer documentation](#).

Payload URL \*  

https://kennethghartman.net/myLMSapi/

Content type  

application/json

Secret  

.....

SSL verification  

🔒 By default, we verify SSL certificates when delivering payloads.

☒ Enable SSL verification ☐ Disable (not recommended)

Which events would you like to trigger this webhook?  

☒ Just the push event.



# *Who owns each security control?*

How can organizational changes that impact control ownership be detected?

# Detecting Control Ownership Changes

1. Document the owner for each security control in a CSV file.
2. Periodically Query AD for specific User Object Attributes for each control owner. Save output as a CSV File with date stamp as part of filename.
3. Perform a “diff” of the most recent two files that contain the AD query results and investigate changes.

## Recommended Attributes:

- department
- manager
- title

## Sample PowerShell Command:

```
Get-ADUser <USERID> -properties title, manager, title
```

**NOTE: The hard part is documenting the security controls and identifying the owner! Coding is the easy part.**



# Security Automation User Stories

As a Security Auditor, I want security automation functionality that will:

- perform a security configuration audit on a configurable schedule.
- detect a deviation from a security configuration audit baseline and generate an event.
- generate emails to inform others of relevant system security information based on a detected event or schedule.
- generate tickets to track the status of actions I need others to take to maintain our security posture based on a detected event or schedule.
- update a ticket based on a security configuration change or other event
- perform an automated task or remediation, based on an event (trigger).




# Declarative vs Imperative

"Imperative programming is like how you do something, and declarative programming is more like what you do."

- **An imperative approach (HOW):** "I see that table located under the window. My friend and I are going to walk over there and sit down."
- **A declarative approach (WHAT):** "Table for two, please."

*"Many (if not all) declarative approaches have some sort of underlying imperative abstraction."* → The **How** is abstracted from the **What**

<https://tylermcginnis.com/imperative-vs-declarative-programming/>



# Declarative Programming & Infrastructure-as-Code

With ***Declarative programming***, Infrastructure-as-code (IaC) scripts can focus on **What** is to be built, rather than **How** to build it.

- AWS CloudFormation
- HashiCorp Terraform

Declarative IaC systems (like CloudFormation and Terraform) can maintain a security configuration when ran repeatedly, thanks to ***idempotence***.

- An **idempotent** operation is one that has no additional effect if it is called more than once with the same input parameters.

# Policy as code is the next phase of infrastructure automation

Infrastructure as Code was the first phase, which enables codification and automation for the four main components of infrastructure — provision, secure, connect, and run. Infrastructure as Code empowers more users to create and manage infrastructure; however, that comes with risks as less experienced users could make significant mistakes that impact business operations. Policy as code limits exposure by codifying business and regulatory policies to ensure infrastructure changes are safe. Together Infrastructure as Code and Policy as code empower users to safely and quickly provision, secure, connect, and run any infrastructure for any application.

[Watch Overview Video](#)

[Read the announcement](#) →



# Policy as Code in Terraform Enterprise

- Policies are enforced in Terraform Enterprise between the plan and apply.
- Policies validate information in the Terraform plan, state, and configuration.

## Examples

- Do not allow resources to be provisioned without tags
- Only provision staging resources in us-west and production resources in us-east
- Do not allow AWS security groups to have egress set to 0.0.0.0

<https://www.hashicorp.com/sentinel/>



# Cloud Custodian

- A rules engine for managing public cloud accounts and resources.
- Manage AWS, Azure, and GCP environments
- Ensure real time compliance to security policies (like encryption and access requirements), tag policies, and cost management.
- Policies are written in YAML files to specify resource type and are constructed from a vocabulary of filters and actions.

## Drives Behavior Change

Notifies users in real-time as they do something wrong.

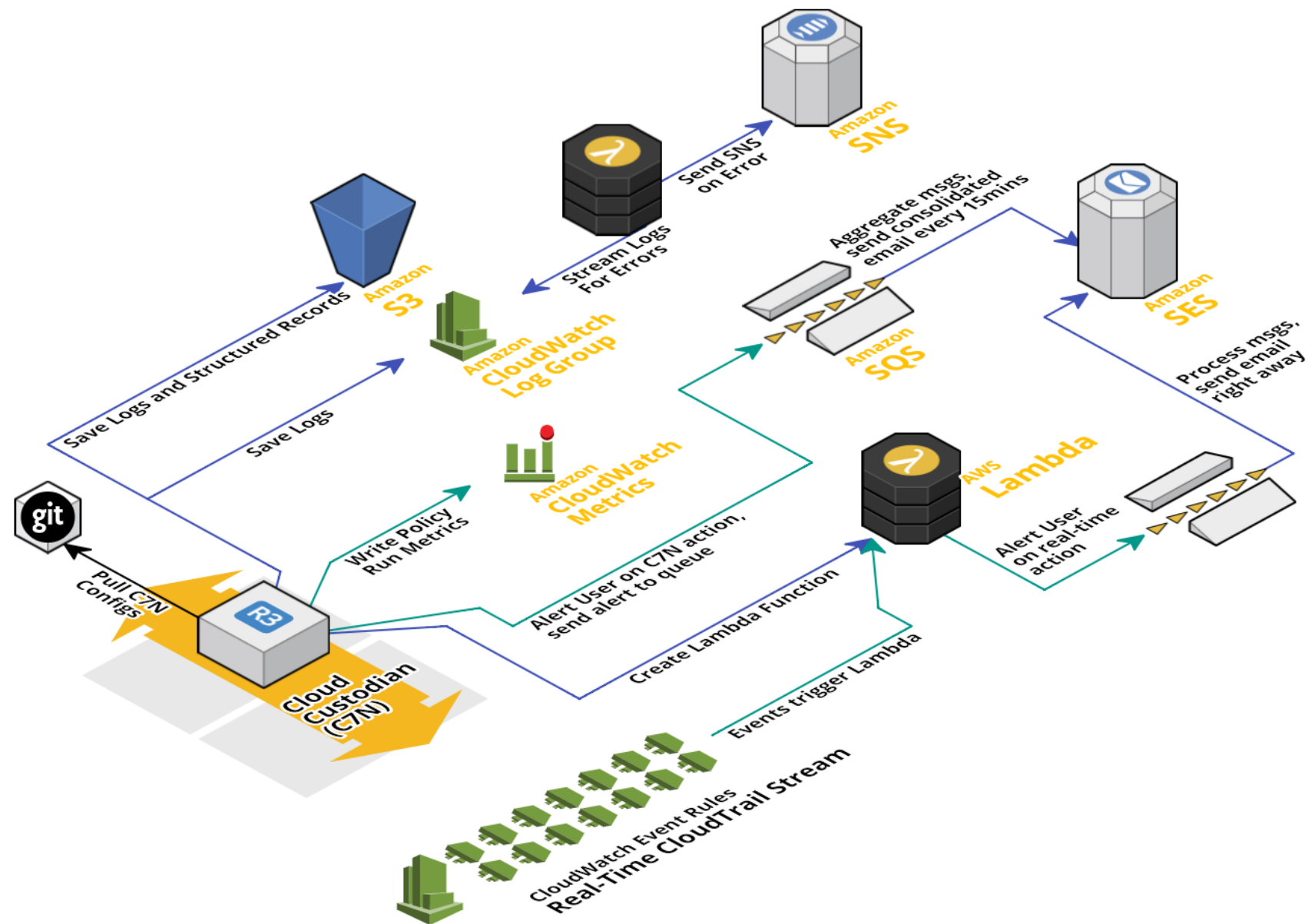


## Drives Compliance

Security/Access Control, Encryption, Backups, etc.

## Drives Cost Savings

Off-hours, Monitoring and Garbage Collection of unused and underutilized resources.





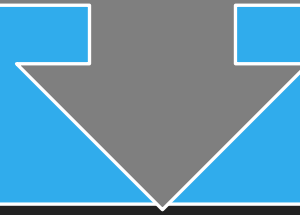
# Example Custodian Policies (AWS)

- Account - Login From Invalid IP Address
- Account - Detect Root Logins
- Account - Service Limit
- AMI - Stop EC2 using Unapproved AMIs
- Block Resources In Non-Standard Regions
- EBS - Create and Manage Snapshots
- EBS - Delete Unencrypted
- EC2 - auto-tag userName on resources
- EC2 - Old Instance Report
- EC2 - Terminate Unpatchable Instances
- Security Groups - Detect and Remediate Violations
- ELB - Delete New Internet-Facing ELBs
- ELB - Delete Unused Elastic Load Balancers
- RDS - Delete Unused Databases With No Connections
- RDS - Terminate Unencrypted Public Instances
- S3 - Configure New Buckets Settings and Standards
- S3 - Block Public S3 Object ACLs
- S3 - Encryption
- Tag Compliance Across Resources (EC2, ASG, ELB, etc.)
- VPC - Flow Log Configuration Check
- VPC - Notify On Invalid External Peering Connections



# Some Thoughts...

Idempotent, declarative systems are nice to have, but not necessary



Terraform, Sentinel, Cloud Custodian, etc. may not be able to enforce every policy, but with creativity, the enforcement of almost every policy is possible somehow



# Guiding Principles

- You cannot secure what you don't know
- Whatever is unmanaged is unlikely secure
- Security works best when designated individuals have clearly defined responsibilities and are held accountable to meet those obligations
- All security requirements (responsibilities) need to be monitored for compliance using automation the moment the requirement becomes effective
- “Perfect is the enemy of good,” therefore use automation and policy-as-code to make iterative improvements per a lightweight change management process



## A Proposed Process for Architecture- Driven DevSecOps Improvement

---

Propose what needs to be done

---

Socialize via “Request for Comments”

---

Design automation to verify/enforce compliance

---

Dev/Sec/Ops/Test Sign-off on the Policy & Code

---

Awareness communications to affected parties

---

Verify/enforce compliance as of Effective Date

---

Measure, Monitor & Refine (Repeat Cycle)



# Questions?