

SANS  
CLOUD  
SECURITY

Cloud Security Automation:  
DevSecOps and Beyond



# Introduction

---

## Frank Kim

- SANS Institute
  - Former CISO
  - Faculty Fellow
  - Curriculum Lead
    - Cloud Security
    - Cybersecurity Leadership
  - Author & Instructor
    - LDR512, LDR514, SEC540
- YL Ventures
  - Former CISO-in-Residence

- Contact

- [fkim@sans.org](mailto:fkim@sans.org)
- [/in/frank-kim](https://www.linkedin.com/in/frank-kim)
- [@fykim](https://twitter.com/fykim)



## Agenda

---



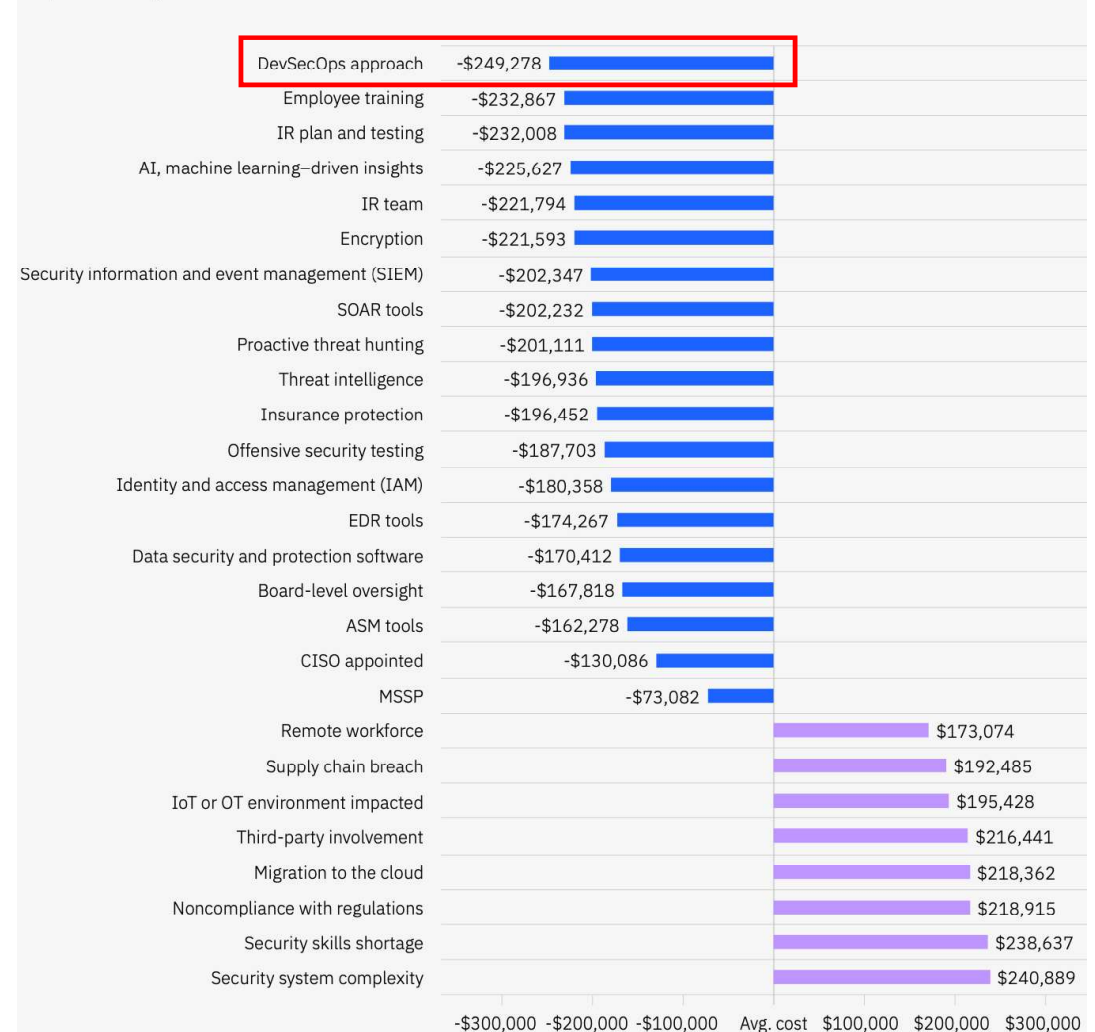
### DevSecOps Foundations

- Automating Code Analysis
- Infrastructure as Code (IaC) Hardening
- Automated Remediation

# Why DevSecOps?

- IBM and Ponemon Cost of Data Breach study
  - Analyzed key factors impacting the cost of a data breach
- DevSecOps approach
  - The #1 recommendation to help reduce the cost of a data breach
  - Resulted in average cost of data breach that was \$249k USD less than the mean cost

Impact of key factors on total cost of a data breach



## DevOps Success Factors

---

- CAMS (or CALMS) is a common lens for understanding DevOps and for driving DevOps change.
- Your organization succeeds when it reaches "CALMS":
  - **C**ulture: people come first
  - **A**utomation: rely on tools for efficiency and repeatability
  - **L**ean: apply Lean engineering practices to continuously learn and improve
  - **M**easurement: use data to drive decisions and improvements
  - **S**haring: share ideas, information, and goals across silos

## DevOps Unicorns – Examples

---

**NETFLIX**

amazon

Etsy

## Security Culture vs. DevOps

DevOps culture conflicts with **traditional security culture**:

- Top-down risk management instead of team-based decision making
- Need to know restrictions versus extended information sharing
- Zero failure versus fail fast and fail forward
- Limiting change: Security is always ready to say "No!"

### DevOps Culture Resources

- The Phoenix Project
- 5 Dysfunctions of a Team
- Lean Enterprise
- Building a DevOps Culture
- The Unicorn Project

## Security Challenges in DevOps

---

Powerful new technologies create opportunities for attackers—and new risks for organizations:

- Weaknesses in the DevOps toolchain can compromise the entire stack.
- Cloud platform misconfigurations can easily allow unauthorized access to data.
- Containers and orchestrators introduce a new attack surface, often not supported by traditional security scanners.
- Microservice-based architectures, new languages, and frameworks compound security guidelines.
- Delivery at the speed of DevOps requires enhanced detection and automated remediation.




## CI/CD Security Risks

The Top 10 CI/CD Security Risks project from OWASP.

**Top 10  
CI/CD  
Security  
Risks**

- CICD-SEC-1 **Insufficient Flow Control Mechanisms**
- CICD-SEC-2 **Inadequate Identity and Access Management**
- CICD-SEC-3 **Dependency Chain Abuse**
- CICD-SEC-4 **Poisoned Pipeline Execution (PPE)**
- CICD-SEC-5 **Insufficient PBAC (Pipeline-Based Access Controls)**
- CICD-SEC-6 **Insufficient Credential Hygiene**
- CICD-SEC-7 **Insecure System Configuration**
- CICD-SEC-8 **Ungoverned Usage of 3rd Party Services**
- CICD-SEC-9 **Improper Artifact Integrity Validation**
- CICD-SEC-10 **Insufficient Logging and Visibility**

 **OWASP**

## CI/CD Security Hardening Guidelines

---

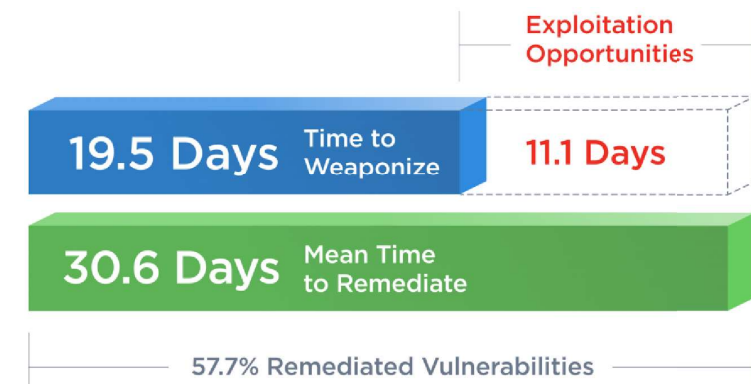
CI/CD security hardening steps vary by provider, but the following controls should be generally followed:

- Restrict control flow into production using branch protections and gated approvals.
- Eliminate service account long-lived credentials to help prevent compromise.
- Limit service account permissions
- Protect the supply chain with allow lists of trusted actions (GH or verified publishers), plugins, and packages.
- Review all changes to workflow files for malicious code execution.
- Patch self-hosted CI/CD runners and software aggressively.
- Include CI/CD audit logs in network and operations monitoring.

## Security as Code: Closing Windows of Exposure

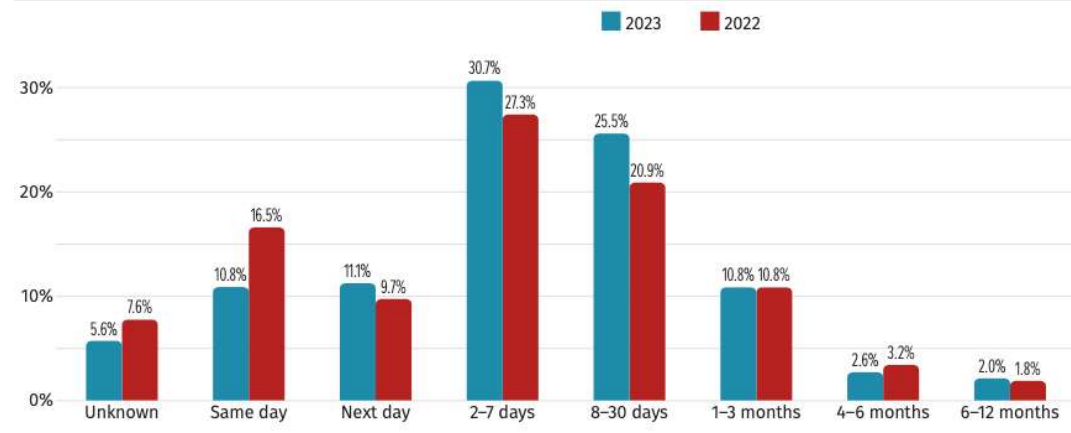
Security teams learning to use the DevOps toolchain can close the vulnerability window faster.

- DevOps encourages people to identify and solve problems together.
- Just-in-time prioritization means patches can be scheduled immediately.
- Continuous Delivery makes rolling out patches fast, cheap, and safe.
- Automated security tests and safety checks (pre/post-deployment) can catch mistakes early.



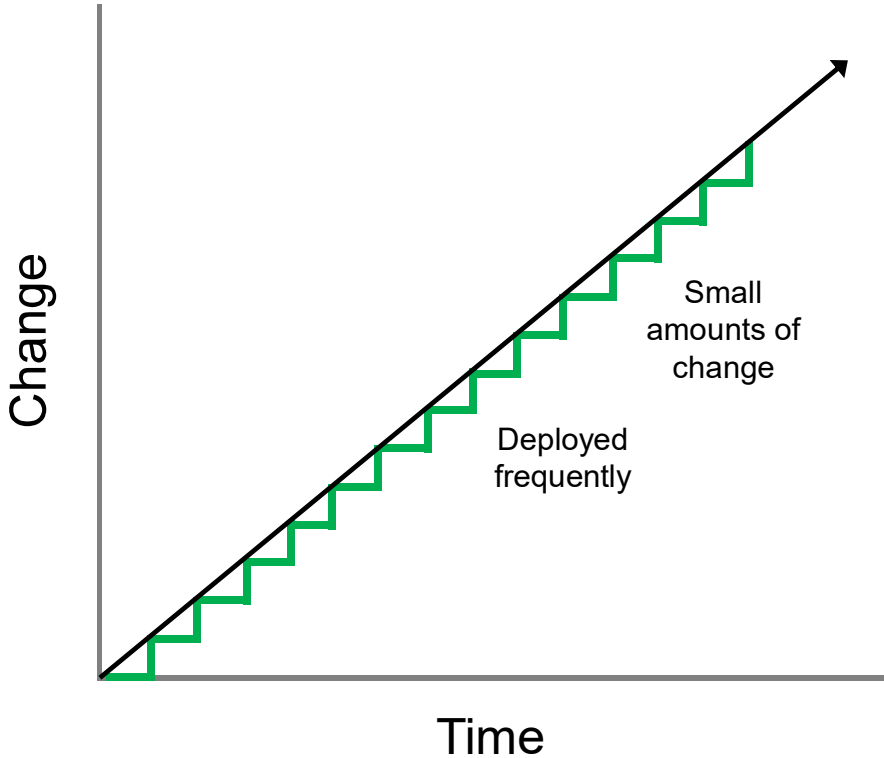
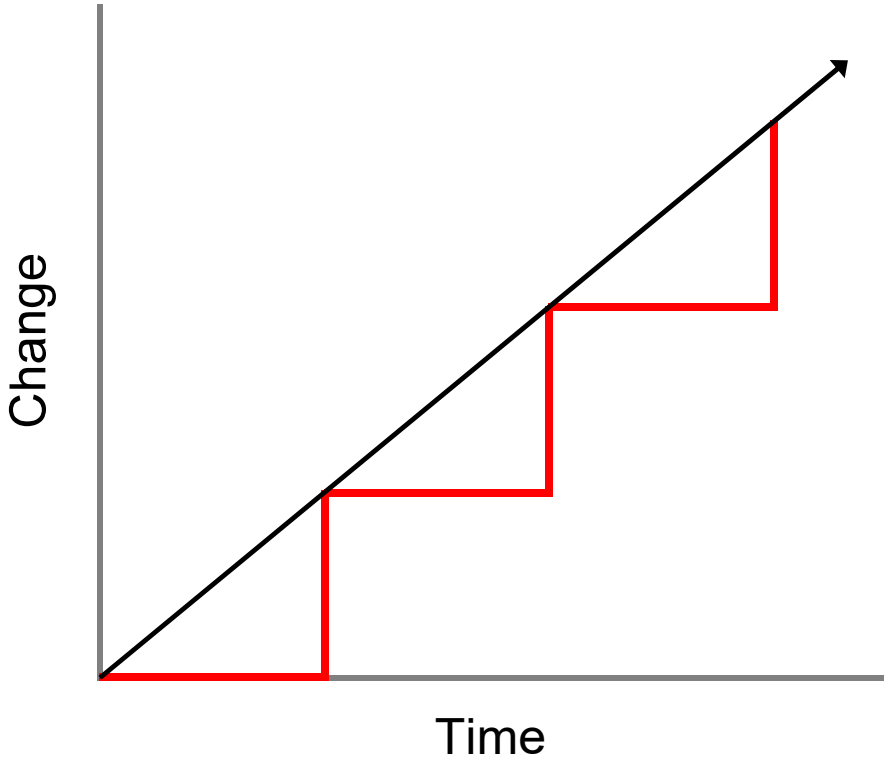
\* 2023 Qualys' TruRisk research report

On average, how long does it take for your organization to patch/resolve critical security risks/vulnerabilities?



\* 2023 SANS DevSecOps Survey

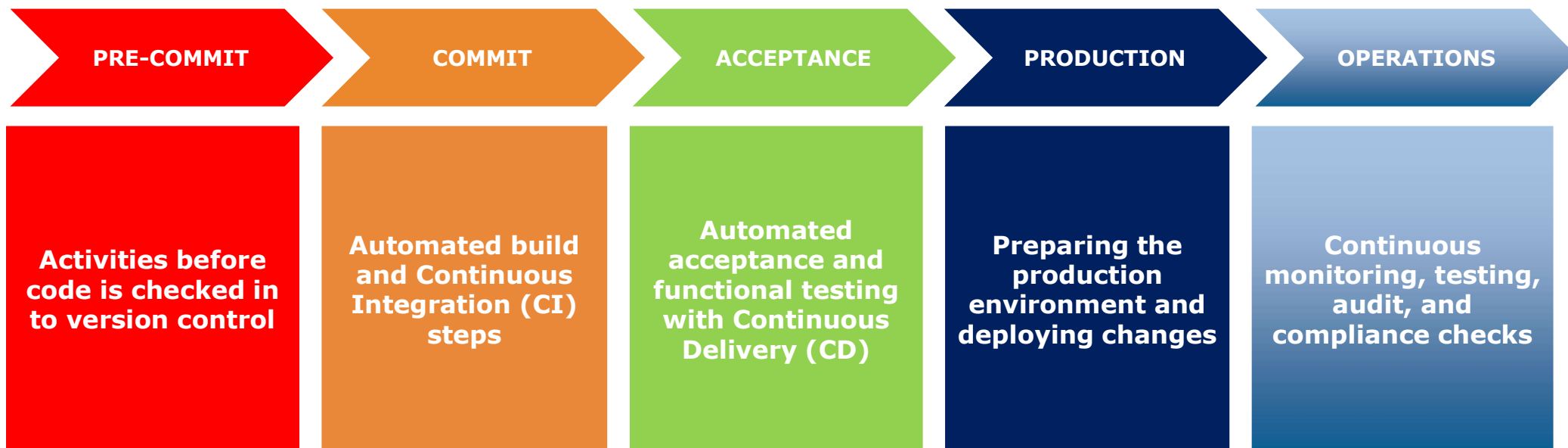
# Making Change Safe



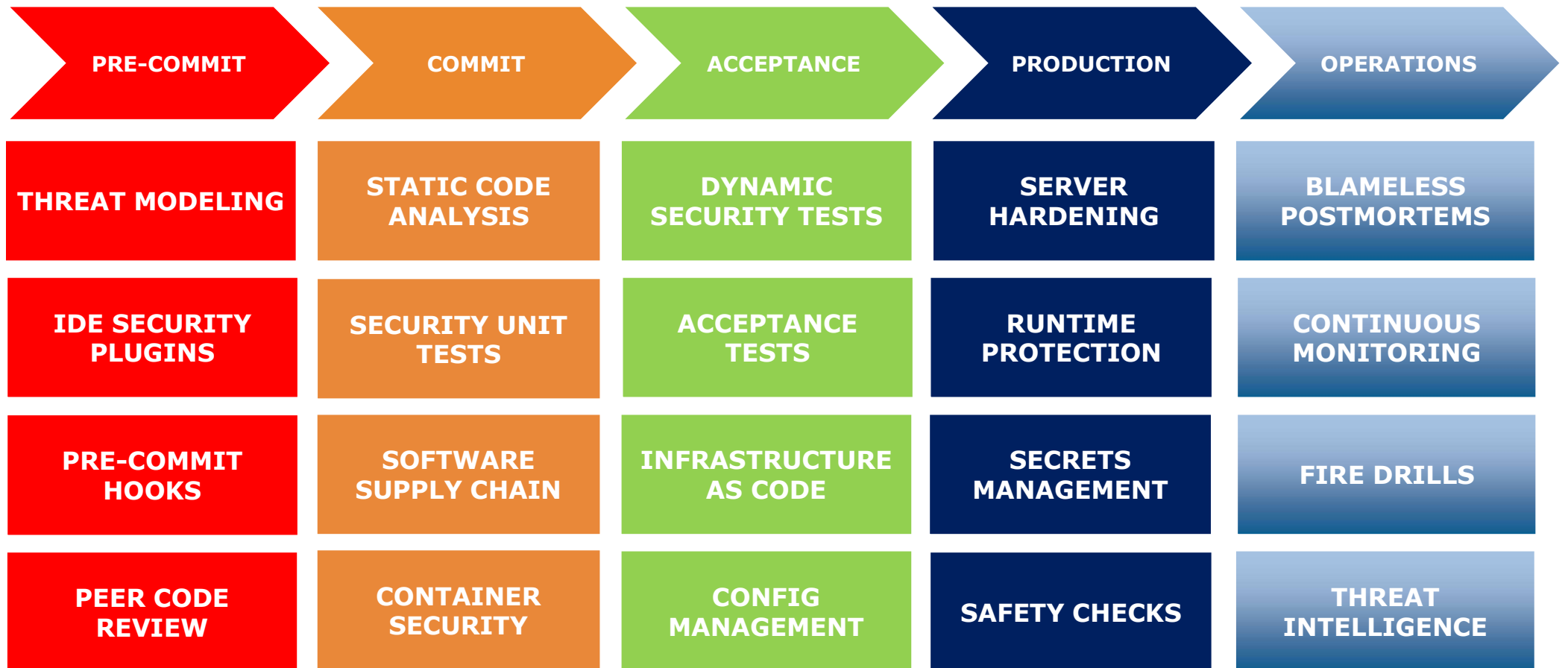
## DevOps Workflow Phases

The DevOps workflow is based on five key phases:

- Manual work done before merging code into a delivery branch
- Automated Continuous Integration (CI) & Continuous Delivery (CD) process

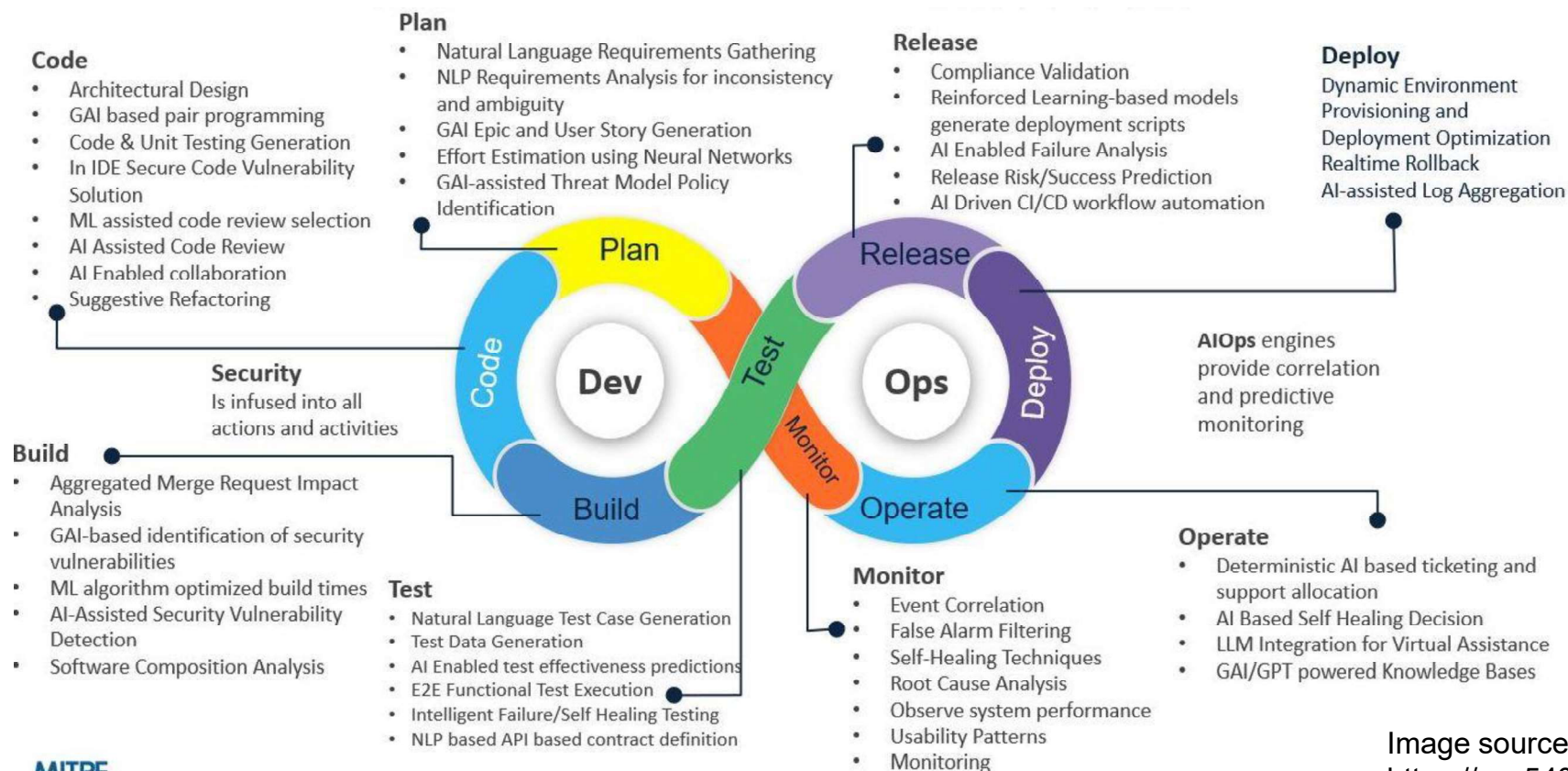


# DevSecOps Critical Security Controls



# DevSecOps and Artificial Intelligence (AI)

Artificial Intelligence (AI), Machine Learning (ML), and Generative AI can help many tasks in the DevSecOps workflow:



## IDE Extensions: GitHub Copilot Example

GitHub Copilot is an AI coding and chat engine for development and security teams:

- Provides contextual auto-completion as you are writing code
- Generates whole functions and unit tests
- Assists secure development efforts analyzing code against vulnerable patterns
- **Caution:** auto-generated code and suggestions are not guaranteed to be bug free. Due diligence required.

The screenshot shows an IDE interface with a file explorer on the left listing various Terraform files under 'DM-INFRASTRUCTURE-AWS', including 'vm.tf'. A chat window on the right is open, displaying a security analysis of the configuration. The analysis includes:

- **metadata\_options** { ... }: This block configures the instance metadata service options.
- **tags = merge(local.tags, { Name = "dm-mgmt-bastion", Product = "Operations" })**: This merges a set of local tags with a set of additional tags and assigns them to the instance.

The chat window also shows a user asking, "Are there any security concerns with this configuration?" and GitHub Copilot responding with a list of six security concerns:

1. **Key Pair Management:** The public key for the `devsecops` key pair is being loaded from a file. Ensure that this file is securely stored and managed, and that the private key is not exposed or shared inappropriately.
2. **User Data:** User data scripts can sometimes contain sensitive information. Make sure that the `bastion_user_data` script does not contain any hard-coded secrets.
3. **Security Group:** The instance is associated with a security group `management_bastion`. Ensure that this security group has strict inbound and outbound rules. Ideally, it should only allow necessary traffic and deny all by default.
4. **Public IP Address:** The instance is set to be associated with a public IP address. This could potentially expose the instance to the internet. Make sure that this is necessary for your use case and that appropriate security measures are in place.
5. **IAM Instance Profile:** The instance is associated with an IAM instance profile `bastion`. Ensure that this IAM role has the least privileges necessary to perform its function.
6. **AMI Source:** The AMI is being sourced from a parameter in the AWS SSM Parameter Store. Ensure that this AMI is from a trusted source.

Remember, security is a broad and complex field, and this analysis is based on the provided code snippet. Always follow the principle of least privilege and regularly review and update your security practices.

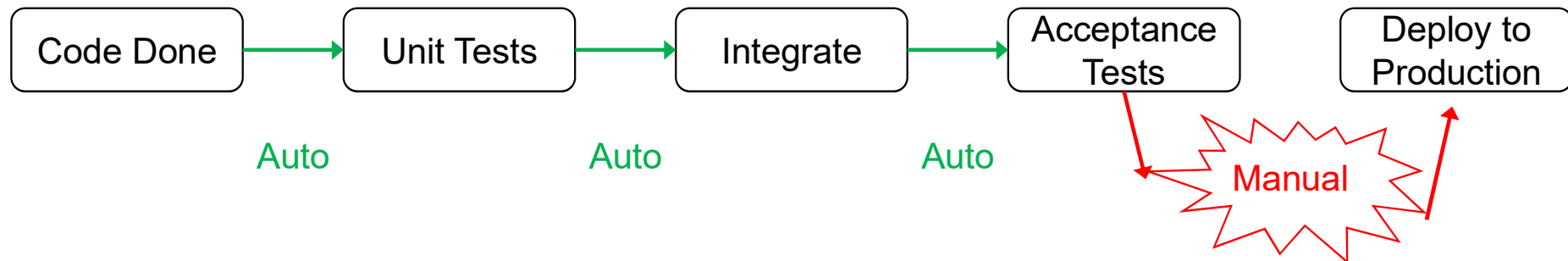
At the bottom of the chat window, a code snippet is visible:

```
34     delete_on_termination = true
35 }
36
```

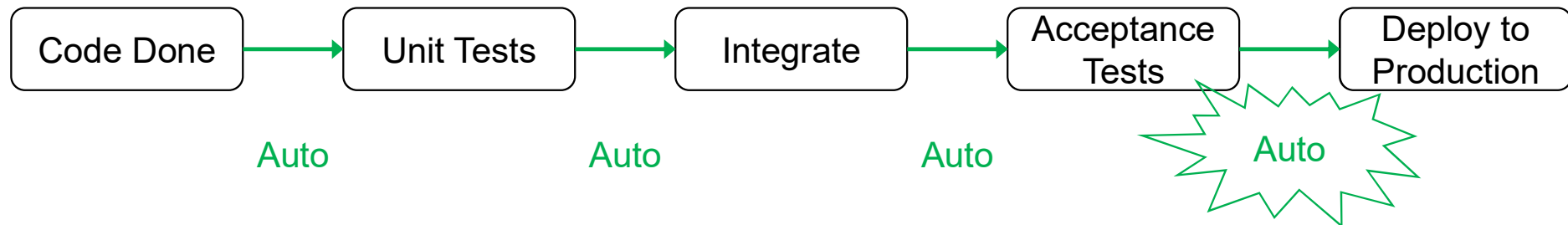


# Continuous Delivery vs. Continuous Deployment

## Continuous Delivery



## Continuous Deployment



## DevOps Toolchain

### CI/CD Tools



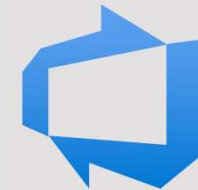
Jenkins



GitHub  
Actions



GitLab  
CI / CD



Azure  
DevOps

### Version Control



GitHub

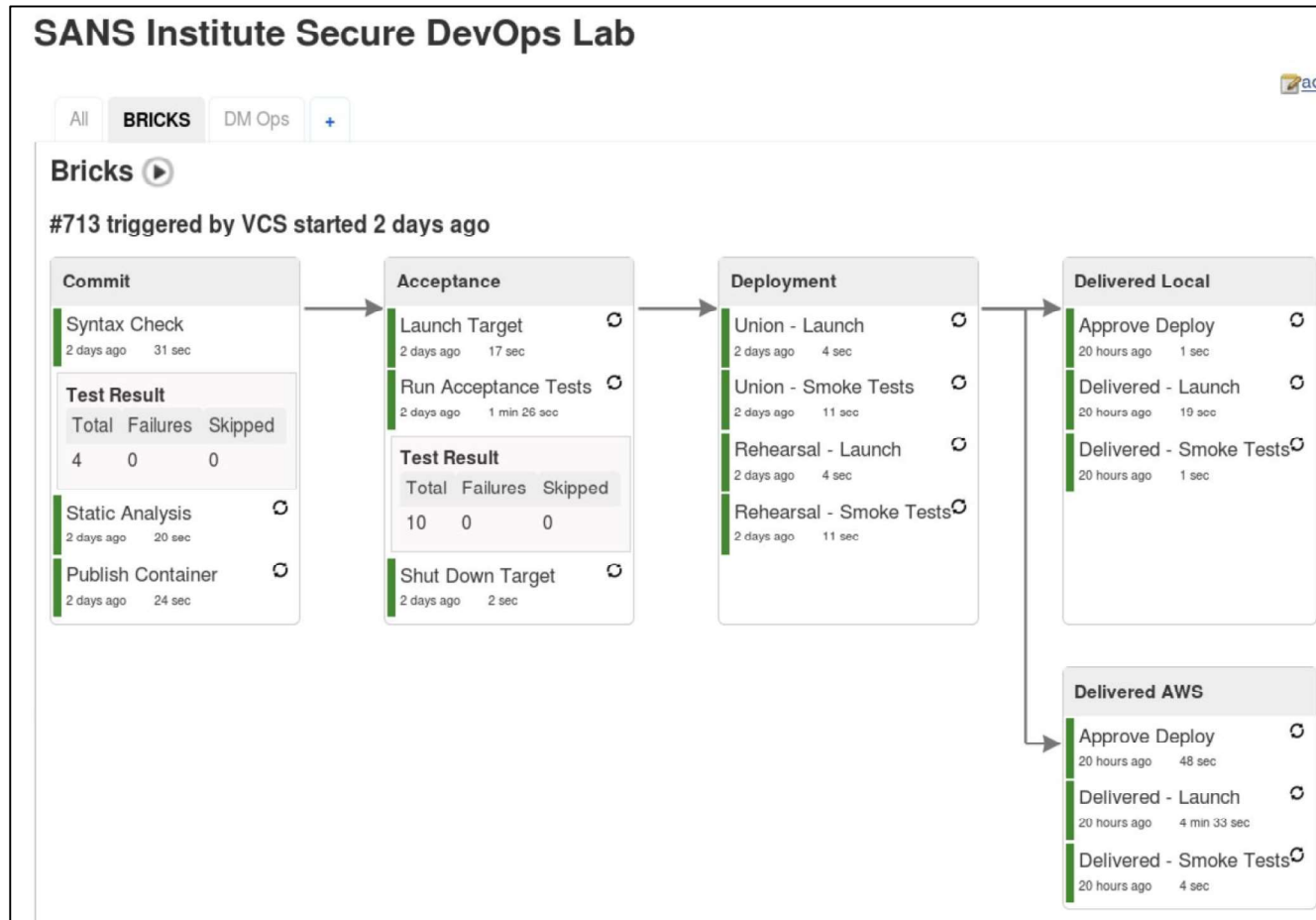


GitLab



Bitbucket

# Jenkins Example

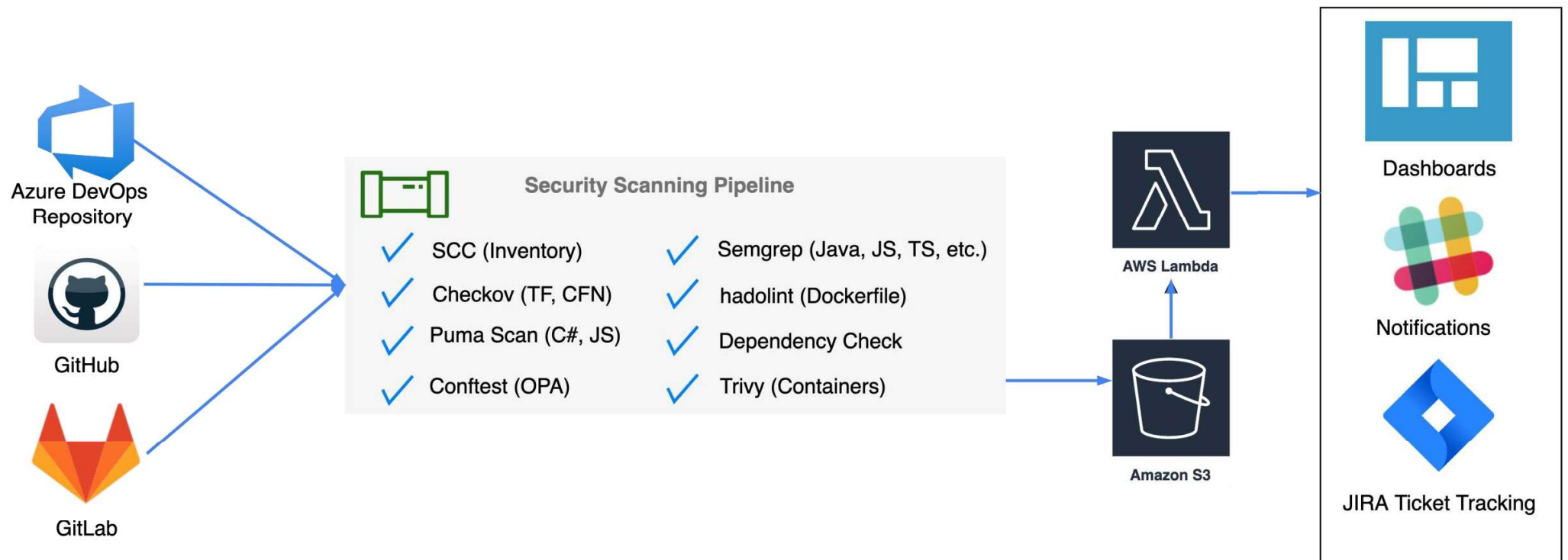


## Agenda

---

- DevSecOps Foundations
- ➔ Automating Code Analysis
- Infrastructure as Code (IaC) Hardening
- Automated Remediation

# DevOps and Security Factory Integration



## Automated Security Testing: Parsing & Displaying Results

Running security tools in CI/CD requires a supported machine-readable output format.

- **xUnit/JUnit**
  - Standard XML schema for reporting pass/fail unit test results
- **Checkstyle**
  - Standard XML schema for reporting static analysis results
- **SARIF** (Static Analysis Results Interchange Format)
  - JSON based schema primarily used for displaying results in GitHub
- **CycloneDX**
  - OWASP schema for software bill of materials and vulnerability exploitability exchange (VEX)
- **SPDX (Software Package Data Exchange)**
  - Open standard for communicating software bill of material provenance, license, and security details
- **JSON**
  - Custom schemas are machine readable, but you have work to do!

## xUnit/JUnit Data Format Example

### 1. Raw JUnit formatted code analysis results:

```
<?xml version="1.0" ?>
<testsuites disabled="0" errors="0"
failures="1" tests="1" time="0.0">
  <testsuite disabled="0" errors="0"
    failures="1" name="semgrep results"
    skipped="0" tests="1" time="0">
    <testcase name="formatted-sql-
      string.formatted-sql-string"
      ...
      file=".../TicketSearchRepository.java"
      line="67">
    <failure type="ERROR" message="Detected a
      formatted string in a SQL statement...">
      rs = stmt.executeQuery(query) ;
    </failure>
    </testcase>
  </testsuite>
</testsuites>
```

### 2. GitLab parsing the JUnit XML file and displaying vulnerability data

#### < semgrep

1 tests                      1 failures                      0 errors                      0% success rate

#### Tests

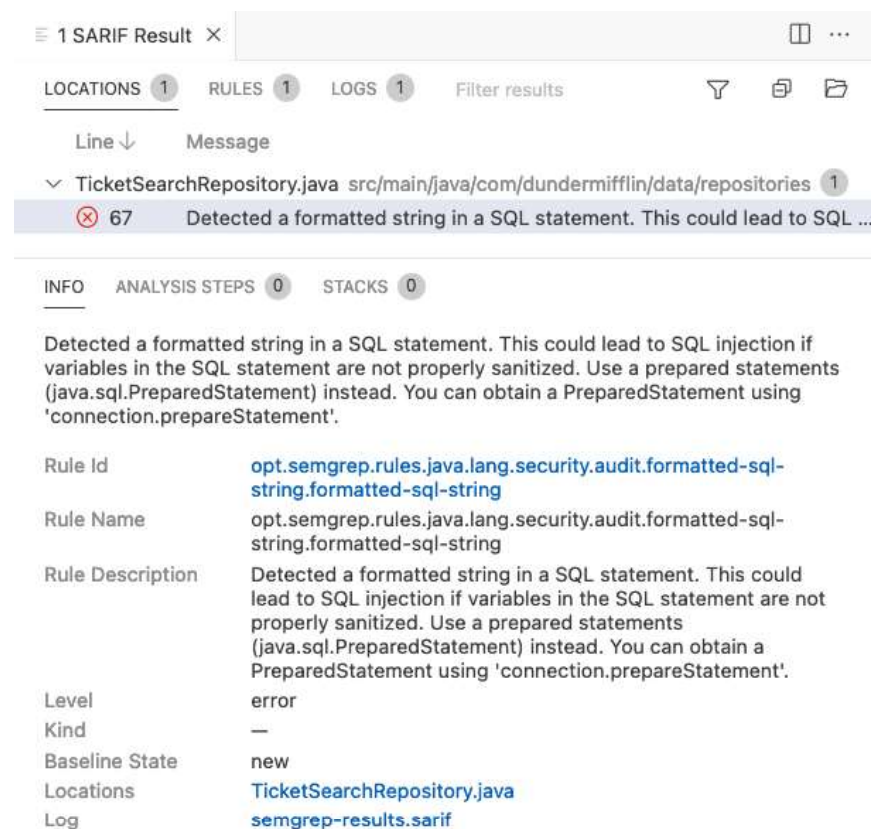
Suite	Name	Filename	Status	Duration
src/main/java/com/dundermifflin/data/repositories/TicketSearchRepository.java	opt.semgrep.rules.java.lang.security.audit.formatted-sql-string.formatted-sql-string	src/main/java/com/dundermifflin/data/repositories/TicketSearchRepository.java		0.00ms

## SARIF Data Format Example

### 1. Raw JSON formatted SARIF code analysis results:

```
{
  "runs": [
    {
      "results": [
        {
          "locations": [
            {
              "physicalLocation": {
                "artifactLocation": {
                  "uri": ".../TicketSearchRepository.java",
                  "uriBaseId": "%SRCROOT%"
                },
                "region": {
                  "startLine": 67
                }
              }
            }
          ],
          "message": {
            "text": "Detected a formatted string in a SQL statement..."
          }
        }
      ]
    }
  ]
}
```

### 2. VSCode displaying SARIF data using the viewer extension:



The screenshot shows the VS Code SARIF viewer extension interface. At the top, it displays "1 SARIF Result". Below this, there are tabs for "LOCATIONS 1", "RULES 1", and "LOGS 1", along with a "Filter results" button and icons for search, refresh, and close. The main area shows a tree view with "TicketSearchRepository.java" expanded, listing a result at line 67: "Detected a formatted string in a SQL statement. This could lead to SQL ...". Below the tree view, there are tabs for "INFO", "ANALYSIS STEPS 0", and "STACKS 0". The "INFO" tab is active, displaying a detailed message: "Detected a formatted string in a SQL statement. This could lead to SQL injection if variables in the SQL statement are not properly sanitized. Use a prepared statements (java.sql.PreparedStatement) instead. You can obtain a PreparedStatement using 'connection.prepareStatement'." Below the message, there is a table of rule details:

Rule Id	<a href="#">opt.semgrep.rules.java.lang.security.audit.formatted-sql-string.formatted-sql-string</a>
Rule Name	opt.semgrep.rules.java.lang.security.audit.formatted-sql-string.formatted-sql-string
Rule Description	Detected a formatted string in a SQL statement. This could lead to SQL injection if variables in the SQL statement are not properly sanitized. Use a prepared statements (java.sql.PreparedStatement) instead. You can obtain a PreparedStatement using 'connection.prepareStatement'.
Level	error
Kind	—
Baseline State	new
Locations	<a href="#">TicketSearchRepository.java</a>
Log	<a href="#">semgrep-results.sarif</a>



## Automated Code Scanning Technology Landscape

Tool support varies widely, depending on your technology stack:

- **Application source code:** open-source tools are available for common languages and frameworks: JavaScript, Java, Python, C#, C/C++, PHP, Ruby on Rails, Android, Objective C, and Go.
- **Software supply chain:** open-source tools are available for application frameworks (npm, Maven, Nuget, PyPI) and scanning container images (more on this later)
- **Configuration management code:** open-source tools are available for Chef, Puppet, Ansible, but are generally limited to lint checks for good coding/correctness. Custom rules and extensions will be required for security coverage.
- **Infrastructure as Code:** open-source tools are available for Terraform, Docker, Kubernetes, CloudFormation, Bicep, and Helm.
- **Limitations:** New languages, frameworks, and technologies have poor (or no) coverage until open-source projects and vendors build in support.

## Semgrep - Secure Code Analysis

Semgrep provides a light-weight, multi-language, extensible static analysis solution.

- Open-source solution built and maintained by Semgrep Inc (fka returntocorp)
- Community driven rules Semgrep Registry contains over 1,000 rules
- Language support includes Go, Java, JavaScript, Python, Ruby, TypeScript, C#, and generic markup (JSON, YAML)
- Cloud offering also supports Secrets and Supply chain scanning
- Supports automation from the CLI, Docker image, and GitHub Actions



## Semgrep - Code Scan Example

- Running a *semgrep* scan against the files in the `/src` directory with the *r2c-security-audit* ruleset
- Scan results are written to **stdout** by default

```
1 $ semgrep scan --config "p/r2c-security-audit" /src
2
3
4 Scan Status
5
6 Scanning 129 files tracked by git with 239 Code rules:
7
8 Language      Rules  Files      Origin      Rules
9 -----
10 <multilang>   13    146        Community   239
11 java         53    54
12 js          25    8
```

## Semgrep – Custom Rule Patterns

Custom rules can be added using the Semgrep playground to test the rule syntax:



The screenshot shows the Semgrep playground interface. At the top, it says "SEMGREP RULE" and has tabs for "Simple" and "Advanced". Below this is a text area labeled "code is:" containing a JSON rule pattern:

```
{ "Id": "S3-Account-Permissions",  
  "Statement": [  
    { "Effect": "Allow",  
      "Action": "s3:*"  
    }  
  ]  
}
```

Below the rule pattern is a "TEST CODE" section with a "Run" button. The test code is a JSON object:

```
1 {  
2   "Version": "2012-10-17",  
3   "Id": "S3-Account-Permissions",  
4   "Statement": [  
5     "Sid": "1",  
6     "Effect": "Allow",  
7     "Principal": {"AWS": ["arn:aws:iam::1234567890:root"]},  
8     "Action": "s3:*",  
9     "Resource": [  
10    ]  
11  ]  
12 }  
13
```

```
1 rules:  
2 - id: s3_wildcard_permissions  
3   pattern: |  
4     { "Id": "S3-Account-  
5       Permissions",  
6       "Statement": [  
7         { "Effect": "Allow",  
8           Action: "s3:*"  
9         }  
10      ]  
11     }  
12 message: Semgrep found a match  
13 severity: WARNING
```

## GitLab CI / CD – Archiving & Displaying Security Test Results

- GitLab *artifacts* are stored with the job for additional processing or downloading after the pipeline is complete.
- GitLab *reports* are specific file types that are parsed and displayed on merge requests, pipeline views, and security dashboards (premium tier only):


```
1  semgrep:
2    image: dmttools/builder_semgrep:stable
3    ...
4    variables:
5      RESULTS_DIR: ./tests/semgrep
6      SARIF_RESULTS: results.sarif
7      JUNIT_RESULTS: results.junit.xml
8    script:
9      - "/bin/bash ./build/bin/semgrep-scan.sh $RESULTS_DIR $SARIF_RESULTS
10         $JUNIT_RESULTS"
11    artifacts:
12      when: always
13      paths:
14        - $RESULTS_DIR/*
15    reports:
16      junit: $RESULTS_DIR/$JUNIT_RESULTS
```

# **Demo #1**

## **Automating Code Analysis**

## Agenda

---

- DevSecOps Foundations
- Automating Code Analysis
-  Infrastructure as Code (IaC) Hardening
- Automated Remediation

# Infrastructure as Code Tooling

## Configuration Management Tools

*To configure & build servers*



ANSIBLE



CHEF

SALTSTACK

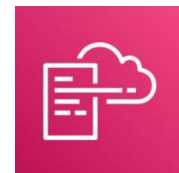


puppet

CFEngine

## Cloud IaC Services

*To configure & deploy cloud infrastructure*



AWS  
CloudFormation



Azure Resource  
Manager (ARM)



GCP Deployment  
Manager



HashiCorp  
Terraform



## AWS CloudFormation Example

### Creating an EC2 instance

```
1 InstancePublic:
2   Type: AWS::EC2::Instance
3   Properties:
4     IamInstanceProfile: !Ref InstanceProfilePhotoReadOnly
5     ImageId: !FindInMap [Images, !Ref "AWS::Region", ecs]
6     InstanceType: "t2.micro"
7     KeyName: "secretKey"
8     SecurityGroupIds:
9       - !Ref SecurityGroupPublic
10    SubnetId: !Ref SubnetPublic
11    UserData:
12      Fn::Base64:
13        !Sub |
14          #!/bin/bash -xe
15          yum update -y
```



## Infrastructure as Code (IaC) Security Scanning Tools

Open-source Infrastructure as Code (IaC) scanning tools:



## Hunt the Bug – Dockerfile Security Review

```
1 FROM mcr.microsoft.com/dotnet/core/sdk:2.2
2
3 ARG ENVIRONMENT=prod
4 ENV ASPNETCORE_ENVIRONMENT=${ENVIRONMENT}
5 ENV CERTIFICATE_PASSPHRASE=mycertpassphrase
6
7 RUN apt-get update && apt-get install -y libssl-dev=1.1*
8
9 COPY cert.conf ./cert.conf
10 RUN openssl req -x509 -nodes -days 3650 -newkey rsa:4096 -keyout app.key
11     -out app.crt -config cert.conf -passin pass:${CERTIFICATE_PASSPHRASE}
12 RUN openssl pkcs12 -export -out app.pfx -inkey app.key -in app.crt
13     -passout pass:${CERTIFICATE_PASSPHRASE}
14
15 USER root
16 WORKDIR /www
17 ENTRYPOINT ["dotnet", "Sans.CreditUnion.API.dll"]
```



## Hunt the Bug – Dockerfile Security Issues

```
1 FROM mcr.microsoft.com/dotnet/core/sdk:2.2
2 ARG ENVIRONMENT=prod
3 ENV ASPNETCORE_ENVIRONMENT=${ENVIRONMENT}
4 ENV CERTIFICATE_PASSPHRASE=mycertpassphrase
5
6 RUN apt-get update && apt-get install -y libssl-dev=1.1*
7
8 COPY cert.conf ./cert.conf
9
10 RUN openssl req -x509 -nodes -days 3650 -newkey rsa:4096 -keyout app.key
11     -out app.crt -config cert.conf -passin pass:${CERTIFICATE_PASSPHRASE}
12 RUN openssl pkcs12 -export -out app.pfx -inkey app.key -in app.crt
13     -passout pass:${CERTIFICATE_PASSPHRASE}
14
15 USER root
16 WORKDIR /www
17 ENTRYPOINT ["dotnet", "Sans.CreditUnion.API.dll"]
```

Incorrect registry?

Unsupported SDK version

Hard coded secret in ENV var

Pinning the package version

Running the application as root



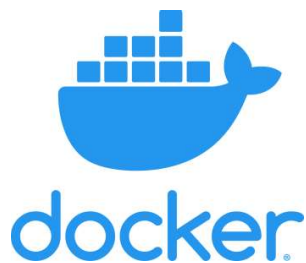
## Container Image Vulnerabilities

- MITRE ATT&CK Containers T1525: Implant Internal Image
  - Images from public registries (e.g., Docker Hub) may contain vulnerabilities or malware—**easy and common attack vector**
- Mitigations:
  - Building inventory of approved base images
  - Downloading base images from trusted suppliers
  - Scanning base images for vulnerabilities
  - Creating a private container registry
  - Signing custom images and storing in a private registry

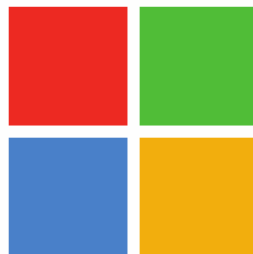


## Container Image Trusted Suppliers

Trusted container base image repositories:



Official Images  
& Verified Publishers



Microsoft Container  
Registry (MCR)



Chainguard  
Images



Platform One  
Iron Bank

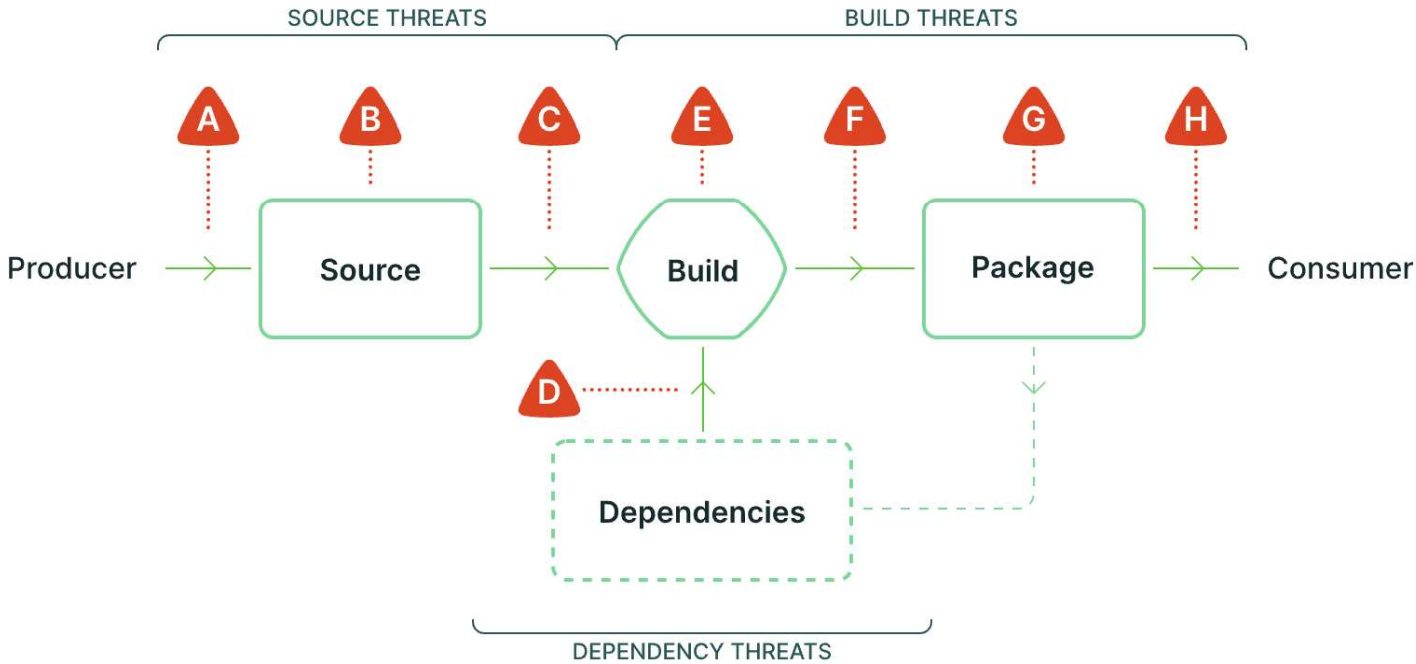


Google Distroless



AWS ECR Public  
Gallery Verified  
Accounts

# Introduction to Supply Chain Security



**SOURCE THREATS**

- A** Submit unauthorized change
- B** Compromise source repo
- C** Build from modified source

**DEPENDENCY THREATS**

- D** Use compromised dependency

**BUILD THREATS**

- E** Compromise build process
- F** Upload modified package
- G** Compromise package registry
- H** Use compromised package

## Supply-chain Levels for Software Artifacts (SLSA)

- SLSA is a specification for describing and incrementally improving supply chain security, established by industry consensus.



Track/Level	Requirements	Focus
Build L0	(none)	(n/a)
Build L1	Provenance showing how the package was built	Mistakes, documentation
Build L2	Signed provenance, generated by a hosted build platform	Tampering after the build
Build L3	Hardened build platform	Tampering during the build



## Software Bill of Materials (SBOM)

- Software Bill of Materials (SBOMs) are a formal, machine-readable inventory of software components and dependencies, information about those components, and their hierarchical relationships.
- OWASP CycloneDX and ISO SPDX are the two most popular formats, and are both available in JSON, YAML, and XML.
- SBOMs enable transparency and awareness, decreasing the time to detect where certain versions of known vulnerable software are in use, improving the awareness what licenses the software we use have, and improving third party assurance through evidence-based, automated information sharing with interested parties.
- However, SBOMs are not a silver bullet, and many generated SBOMs today are incomplete. Caution must be used in completely relying on SBOM data.

# Demo #2

## Infrastructure as Code Hardening

## Agenda

---

- DevSecOps Foundations
- Automating Code Analysis
- Infrastructure as Code (IaC) Hardening



Automated Remediation

## Cloud Custodian

- Open-source tool from Capital One to manage cloud environments:
  - CNCF project with policies for security and cost management
  - Supports AWS, Azure, and GCP
  - Created by Kapil Thangavelu @kapilvt
- Example policies:
  - Detect root logins and logins from invalid IPs
  - Block resources in non-standard regions
  - Configure ELB TLS ciphers and protocols
  - Configure settings and block public S3 object ACLs
  - Detect and remediate SecurityGroup violations



## Cloud Custodian: Azure Modes

- `azure-periodic`
  - Creates an Azure Function that is triggered at specified times
  - Based on user defined cron interval
- `pull`
  - Executes Cloud Custodian policy wherever it is run
- `azure-event-grid`
  - Creates an Event Grid triggered Azure Function
  - Allows you to apply Custodian policies when events occur in Azure

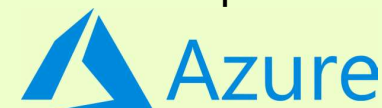


## Cloud Custodian: Policy File for Azure

```
1 policies:
2   - name: security-groups-remediate
3     resource: azure.networksecuritygroup
4     mode:
5       type: azure-event-grid
6     filters:
7       - type: ingress
8         exceptPorts: '22,443,3443,8080,8443'
9         match: 'any'
10        access: 'Allow'
11    actions:
12      - type: close
13        exceptPorts: '22,443,3443,8080,8443'
14        direction: 'Inbound'
```



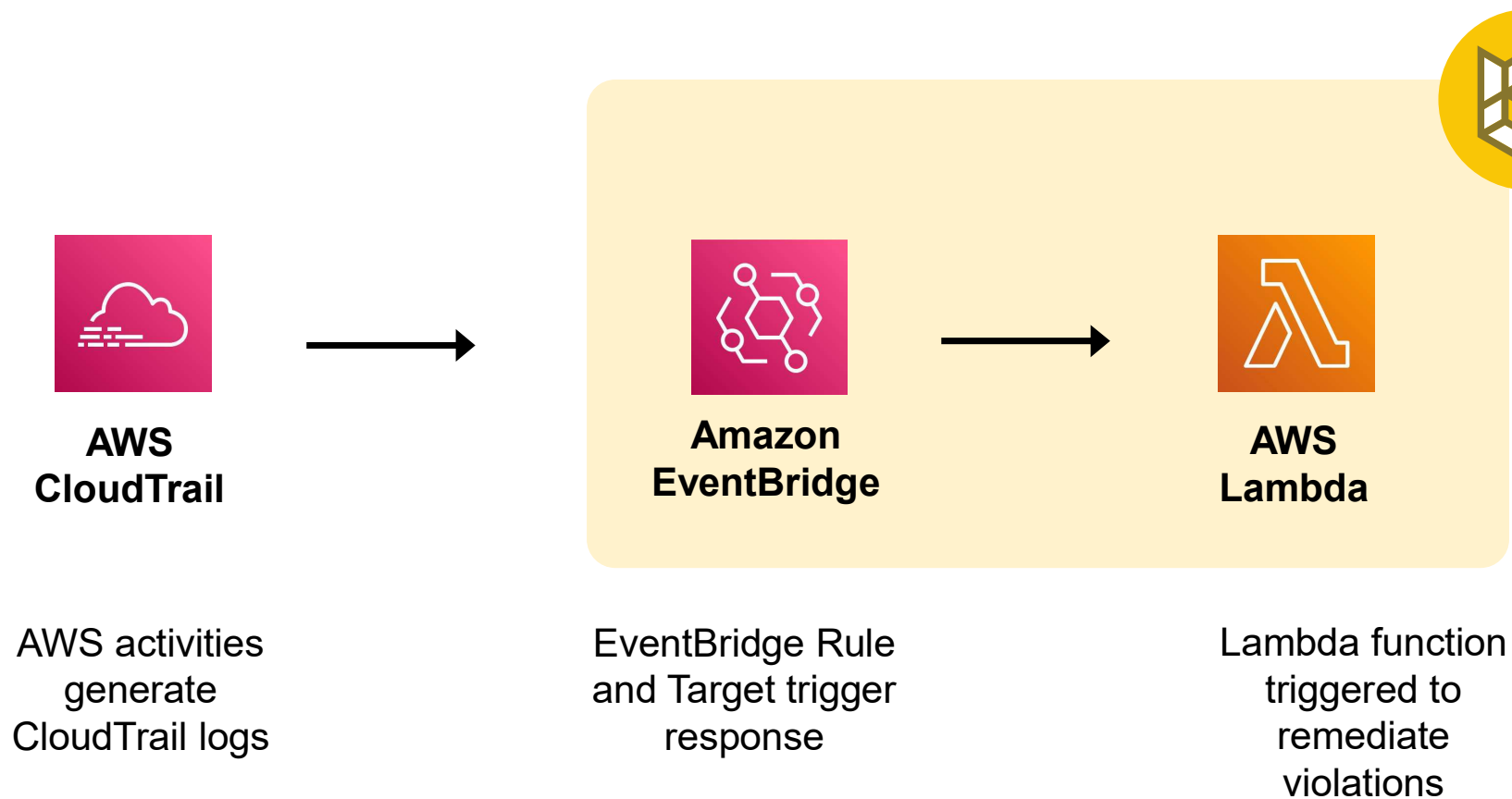
+



Detect when inbound rule is enabled on any ports except these

Close any open ports except these

## Cloud Custodian: Remediation Workflow



## Cloud Custodian: Policy File for AWS

```

policies:
- name: high-risk-security-groups-remediate
  resource: security-group
  description: |
    Remove any rule from a security group that allows 0.0.0.0/0 or ::/0 (IPv6) ingress
    and notify the user who added the violating rule.
  mode:
    type: cloudtrail
    events:
      - source: ec2.amazonaws.com
        event: AuthorizeSecurityGroupIngress
        ids: "requestParameters.groupId"
  filters:
    - or:
      - type: ingress
        Cidr:
          value: "0.0.0.0/0"
      - type: ingress
        CidrV6:
          value: "::/0"
  actions:
    - type: remove-permissions
      ingress: matched

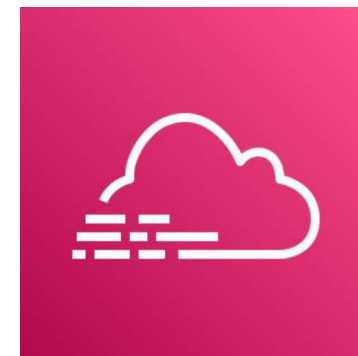
```





## Cloud Custodian: CloudTrail Log

```
"eventTime": "2019-04-09T16:53:39Z",
"eventSource": "ec2.amazonaws.com",
"eventName": "AuthorizeSecurityGroupIngress",
"awsRegion": "us-west-2",
"sourceIPAddress": "67.169.115.247",
"userAgent": "signin.amazonaws.com",
"requestParameters": {
  "groupId": "sg-099675a2740d77207",
  "ipPermissions": {
    "items": [
      {
        "ipProtocol": "tcp",
        "fromPort": 0,
        "toPort": 65535,
        "groups": {
        },
        "ipRanges": {
          "items": [
            {
              "cidrIp": "0.0.0.0/0"
            }
          ]
        }
      }
    ]
  }
}
```



**AWS  
CloudTrail**

# Cloud Custodian: EventBridge Event Rule

## Event Source

Build or customize an Event Pattern or set a Schedule to invoke Targets.

Event Pattern ⓘ  Schedule ⓘ

Build event pattern to match events by service ▾

Service Name

Event Type

For AWS API call events, CloudWatch Events supports the same read/write APIs as CloudTrail does. Read-only APIs, such as those that begin with **List**, **Get**, or **Describe** are not supported by CloudWatch Events. [See more details](#) about which services are supported by CloudTrail.

Any operation  Specific operation(s)

<input type="text" value="AuthorizeSecurityGroupIngress"/>	<input type="button" value="✕"/>
<input type="text" value="AuthorizeSecurityGroupEgress"/>	<input type="button" value="✕"/>
<input type="text" value="RevokeSecurityGroupEgress"/>	<input type="button" value="✕"/>
<input type="text" value="RevokeSecurityGroupIngress"/>	<input type="button" value="✕"/>



**Amazon  
EventBridge**

# Cloud Custodian: EventBridge Event Log

16:54:05	START RequestId: c8414088-083f-49cf-a3b6-bf4c47ca4534 Version: \$LATEST	
16:54:05	[INFO] 2019-04-09T16:54:05.928Z c8414088-083f-49cf-a3b6-bf4c47ca4534 Processing event	EventBridge triggers less than a minute after the bad event
16:54:05	"eventTime": "2019-04-09T16:53:39Z",	
16:54:05	"eventSource": "ec2.amazonaws.com",	
16:54:05	"eventName": "AuthorizeSecurityGroupIngress",	Event name is AuthorizeSecurityGroupIngress
16:54:05	"awsRegion": "us-west-2",	
16:54:05	"sourceIPAddress": "67.169.115.247",	
16:54:05	"userAgent": "signin.amazonaws.com",	
16:54:05	"requestParameters": {	
16:54:05	"groupId": "sg-099675a2740d77207",	SecurityGroup that was updated
16:54:05	"ipPermissions": {	
16:54:05	"items": [	
16:54:05	{	
16:54:05	"ipProtocol": "tcp",	
16:54:05	"fromPort": 0,	
16:54:05	"toPort": 65535,	
16:54:05	"groups": {},	
16:54:05	"ipRanges": {	
16:54:05	"items": [	
16:54:05	{	
16:54:05	"cidrIp": "0.0.0.0/0"	Allows all IPs
16:54:05	}	Event invokes a RemovePermission action
16:54:06	[DEBUG] 2019-04-09T16:54:06.721Z c8414088-083f-49cf-a3b6-bf4c47ca4534 metric:ResourceCount Count:1 policy:high-risk-security-groups-remediate restype:secu	
16:54:06	[INFO] 2019-04-09T16:54:06.721Z c8414088-083f-49cf-a3b6-bf4c47ca4534 Invoking actions [c7n.resources.vpc.RemovePermissions object at 0x7f6ad9c05fd0>, <c7	
16:54:06	[INFO] 2019-04-09T16:54:06.721Z c8414088-083f-49cf-a3b6-bf4c47ca4534 policy: high-risk-security-groups-remediate invoking action: removepermissions resources:	
16:54:06	[INFO] 2019-04-09T16:54:06.926Z c8414088-083f-49cf-a3b6-bf4c47ca4534 policy: high-risk-security-groups-remediate invoking action: notify resources: 1	

# Cloud Custodian: EventBridge Event Target

## Targets

Select Target to invoke when an event matches your Event Pattern or when schedule is triggered.

**Lambda function** ▼ ✕

Function\* custodian-high-risk-security-groups-remediate ▼

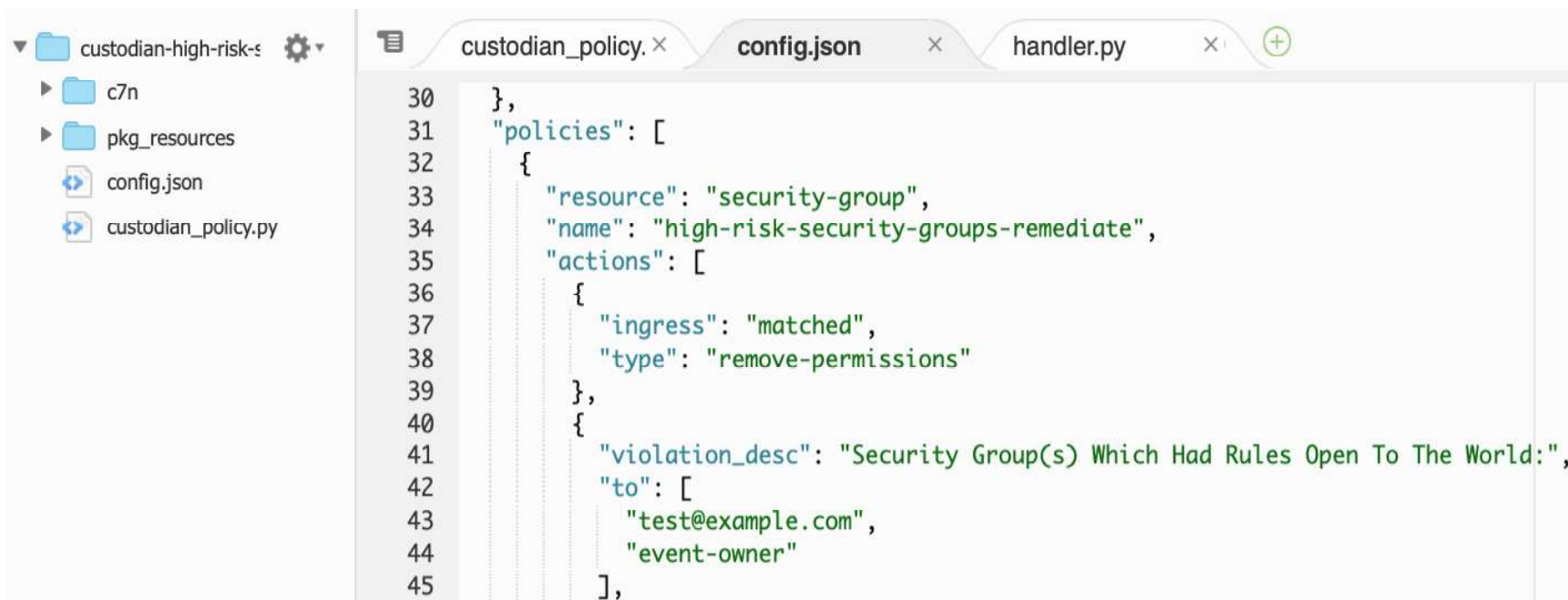
- ▶ Configure version/alias
- ▶ Configure input

⊕ Add target\*



**Amazon  
EventBridge**

# Cloud Custodian: Lambda Function



```
30 },
31 "policies": [
32   {
33     "resource": "security-group",
34     "name": "high-risk-security-groups-remediate",
35     "actions": [
36       {
37         "ingress": "matched",
38         "type": "remove-permissions"
39       },
40       {
41         "violation_desc": "Security Group(s) Which Had Rules Open To The World:",
42         "to": [
43           "test@example.com",
44           "event-owner"
45         ],

```



**AWS  
Lambda**

# **Demo #3**

## **Automated Remediation**

## Summary

---

- DevSecOps Foundations
- Automating Code Analysis
- Infrastructure as Code (IaC) Hardening
- Automated Remediation

# CLOUD ACE JOURNEYS

[sans.org/cloud-security/ace](https://sans.org/cloud-security/ace)

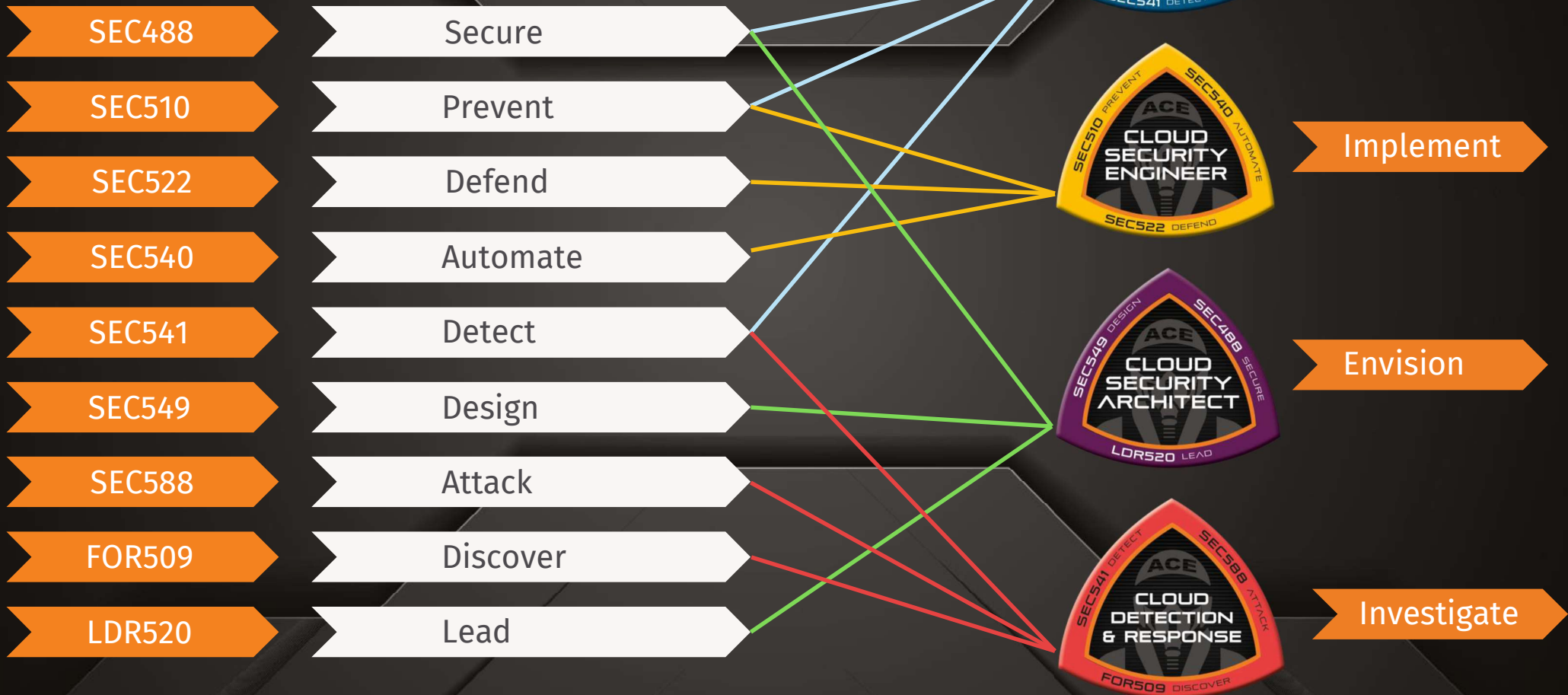


CLOUD  
SECURITY

SANS



# Cloud Security Wireframe





# SANS CLOUD SECURITY

# CURRICULUM ROADMAP

## Baseline

SEC  
388

### Introduction to Cloud Computing and Security

Ground school for cloud security

## Foundational Security Techniques

SEC  
488

### Cloud Security Essentials | GCLD

License to learn cloud security.



## Security Management

MGT  
520

### Leading Cloud Security Design and Implementation

Chart your course to cloud security.

## Core

SEC  
510

### Public Cloud Security: AWS, Azure, and GCP | GPCS

Multiple clouds require multiple solutions.



SEC  
540

### Cloud Security and DevSecOps Automation | GCSA

The cloud moves fast. Automate to keep up.



SEC  
541

### Cloud Security Attacker Techniques, Monitoring & Threat Detection | GCTD

Attackers can run but not hide. Our radar sees all threats.



SEC  
549

### Enterprise Cloud Security Architecture

Design it right from the start.

## Specialization

SEC  
522

### Application Security: Securing Web Apps, APIs, and Microservices | GWEB

Not a matter of "if" but "when." Be prepared for a web attack. We'll teach you how.



SEC  
588

### Cloud Penetration Testing | GCPN

Aim your arrows to the sky and penetrate the cloud.



FOR  
509

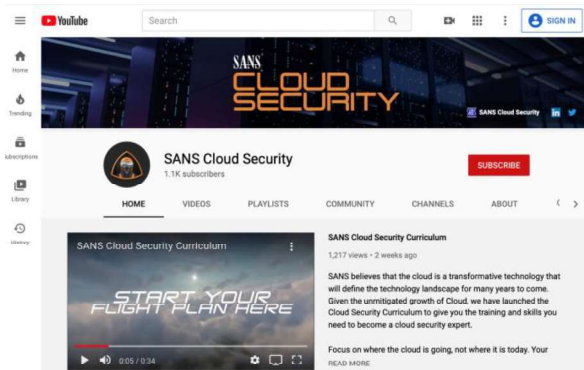
### Enterprise Cloud Forensics and Incident Response | GCFR

Find the storm in the cloud.



# SANS CLOUD SECURITY

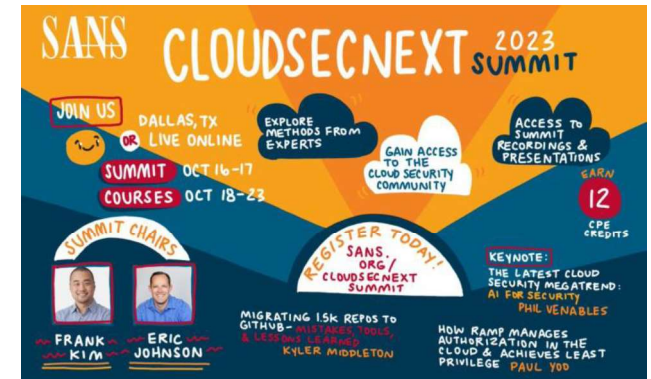
## Free Resources



Webcasts



Workshops



Summits



Cloud Ace Podcast



Surveys, Papers, Posters



[sans.org/cloud-security](https://sans.org/cloud-security)

# Questions?

**Frank Kim**

fkim@sans.org

/in/frank-kim

@fykim

*Material based on SANS SEC540*