



DevSecOps with TDS

Amazon Web Services
Solutions Architect, 주성식

Tuesday, August 31, 2021



Agenda

- DevSecOps
- Test Driven Security (TDS)
- Compliance as Code
- CICD Pipeline Security
- DEMO
- Closing

DevSecOps



성숙한 DevOps 환경에서
보안은 모두의 공동 책임입니다.

DevOps

DevOps는 장벽을 제거하는 것입니다.

DevOps에서는 팀이 함께 작업하여 개발자의 생산성과 운영의 안정성 모두를 최적화합니다.

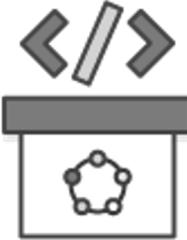
도구보다 프로세스, 프로세스보다 사람

DevOps

- DevOps의 이점



지속적
통합



지속적
전달



마이크로 서비스



코드형
인프라



모니터링 및
로깅



소통 및
협업



여러분의 시스템은
어느정도 수준으로 안전하게 보호되고 있습니까?

DevSecOps

- 다시 생각해 보는 보안..

"배는 항구에서 **안전**하지만,
그것이 배가 만들어진 목적은 아니다."



Koei, 대항해시대2 (1993)

"A ship in harbor is safe, but that is not what ships are built for. " by John A. Shedd (1850 ~ 1926)

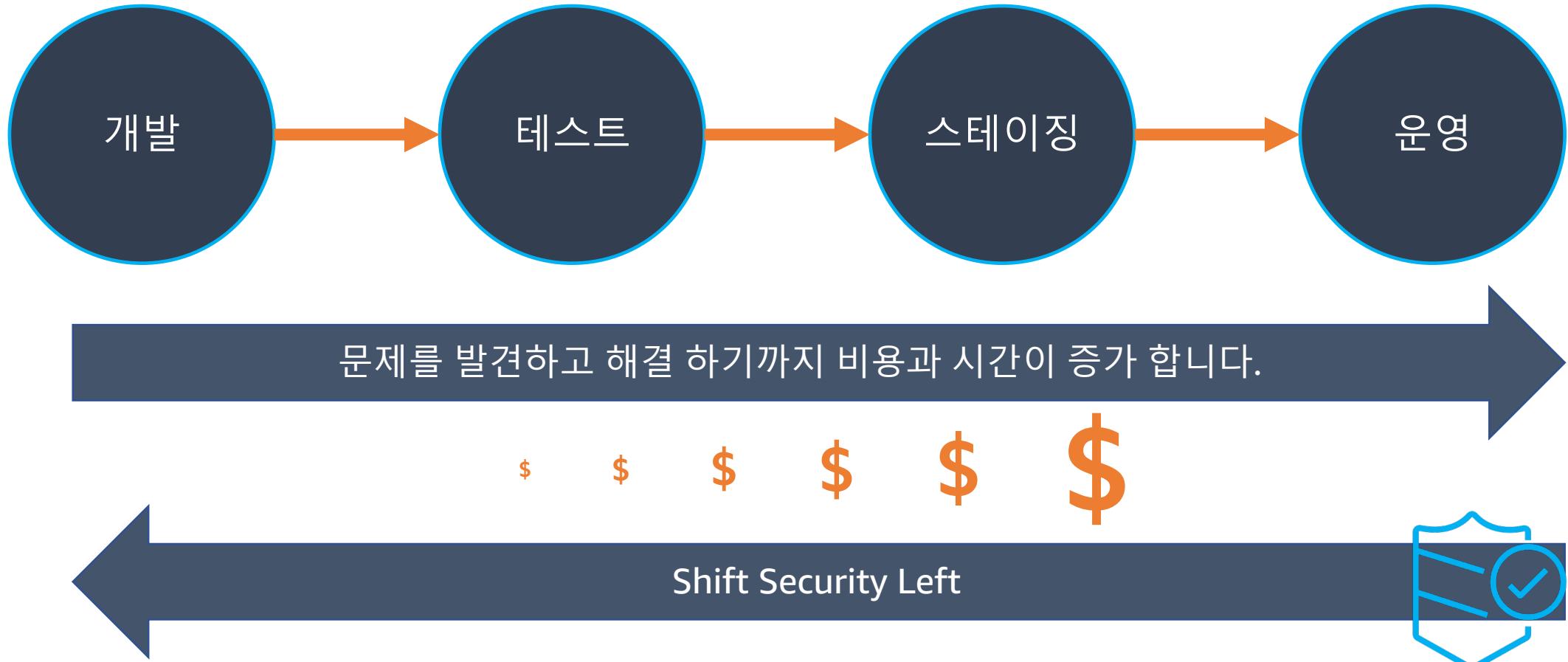
DevSecOps

데브옵스 = 수명 주기를 가속화하는 효율성

DevSecOps = 수명 주기 속도에 영향을 주지 않고 빌딩 블록을 검증

DevSecOps

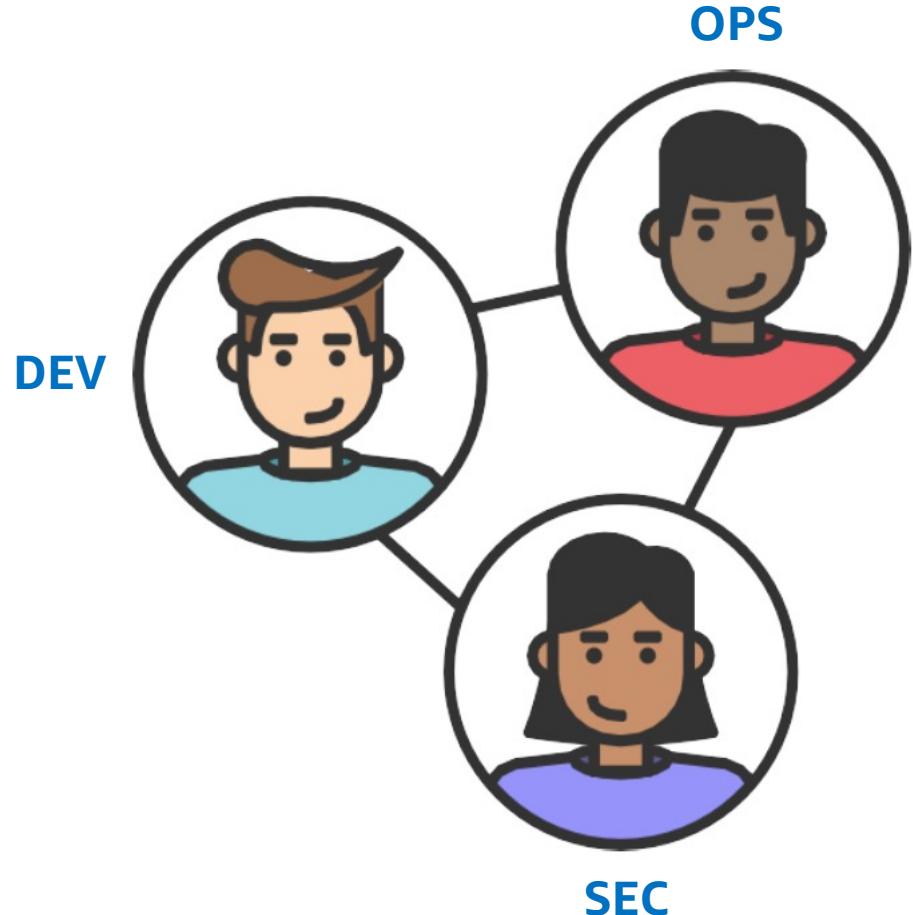
- Shift (All the Way) Left



DevSecOps

- DevOps 환경에서의 보안

- DevOps 파이프라인의 속도를 늦추지 않아야 함
- 보안의 여러 구성 요소를 Code로 표현하는 것으로부터 출발
- 자동화된 도구와 프로세스
- CI/CD 의 일환으로 코드 및 아티팩트를 검증 및 감사 Control 구현

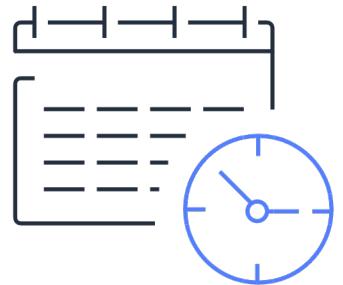


DevSecOps

- DevOps 조직에 DevSecOps를 도입하는 방법
 - 개발 환경 전체에서 견고한 DevOps 기반을 구축합니다.
 - 개발 팀과 보안 팀 간의 협업을 촉진하여 처음부터 설계에 보안을 포함시킵니다.
 - 지속적 보안 테스트를 자동화하고 지속적 통합/지속적 개발 파이프라인에 빌드합니다.
 - 설계 상태에서 벗어나는 편차를 실시간으로 모니터링하여 보안 및 규정 준수를 포함하도록 모니터링을 확장합니다.
 - 미준수로 표시된 리소스의 알림, 자동화된 재조정 또는 격리를 활성화해야 합니다.

DevSecOps

- **Tenets of DevSecOps**



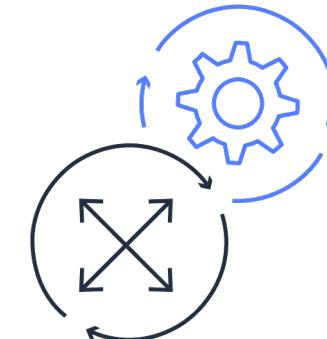
좀 더 일찍 테스트



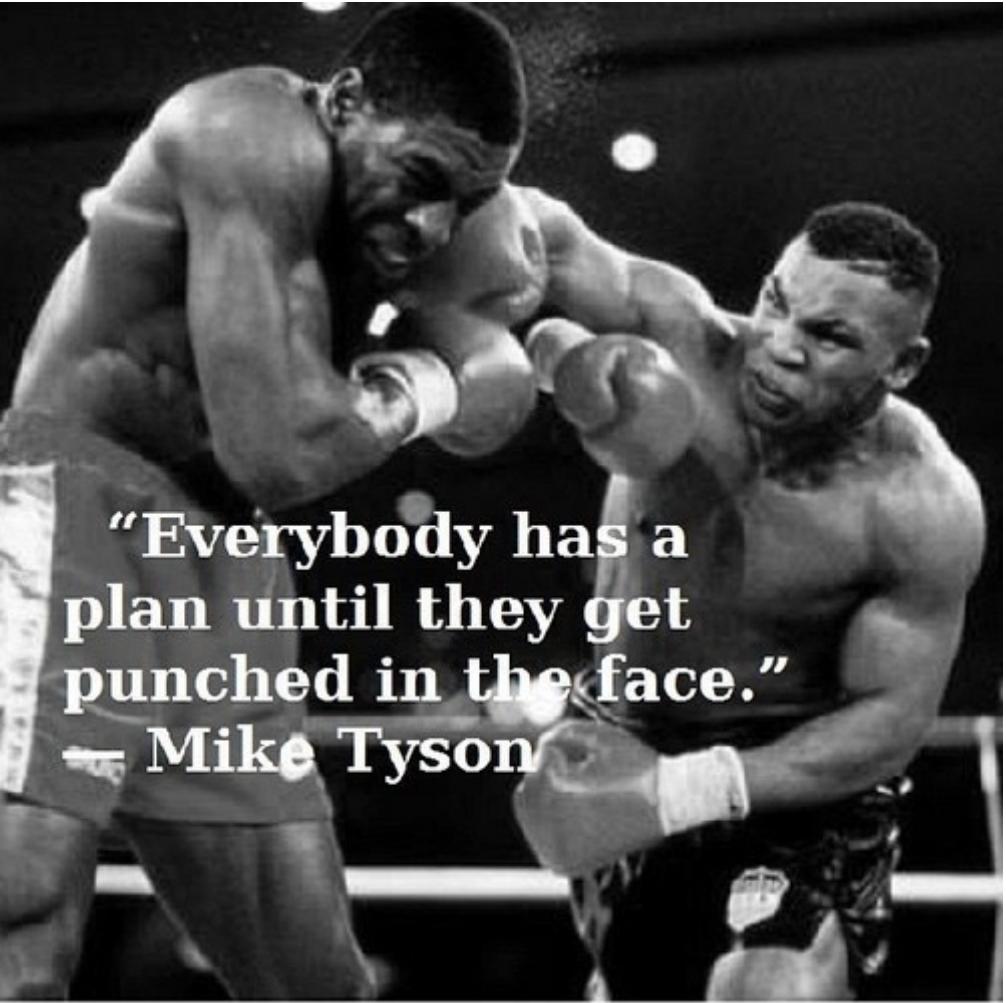
발견된 보안사항에 대한
자동화된 컨트롤



사고예방을 위한 우선순위 지정



자동화, 자동화, 자동화!



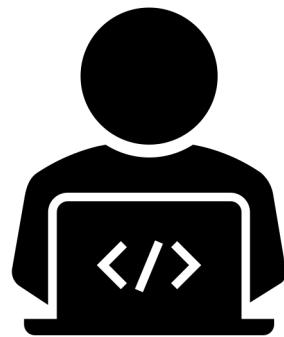
**"Everybody has a
plan until they get
punched in the face."**

— Mike Tyson

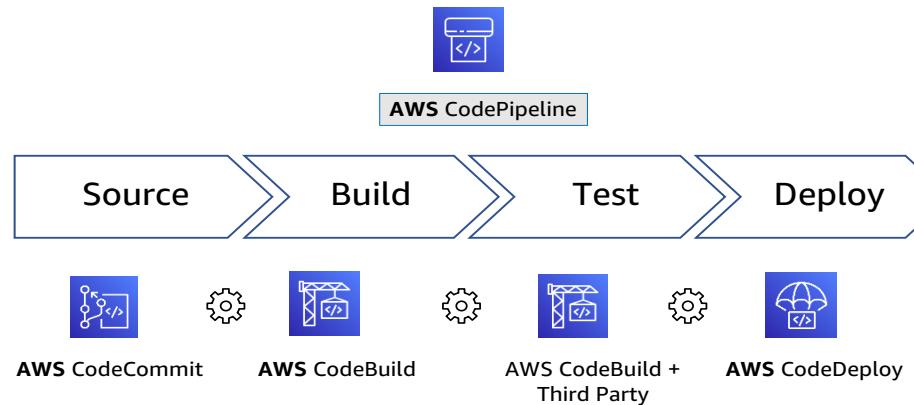
Do you have a plan ?

DevOps의 사이클

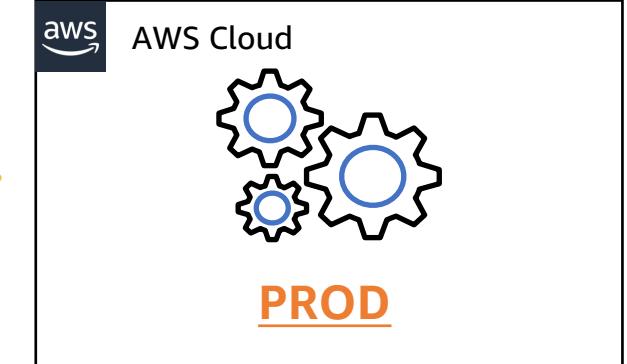
Development



CI/CD

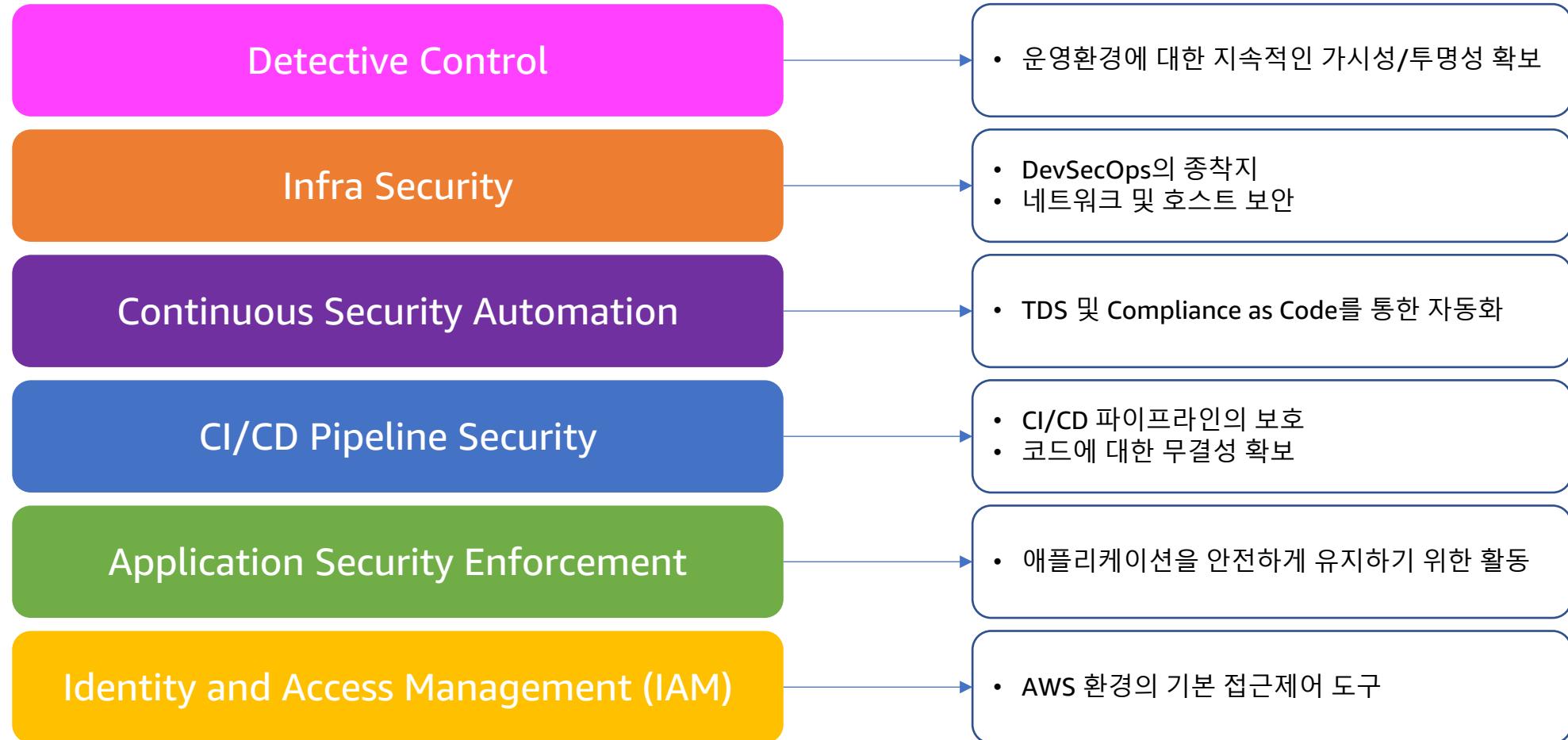
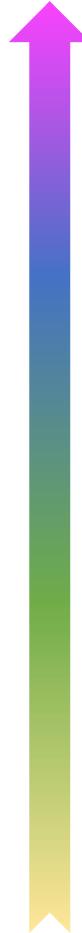


Operation



Security

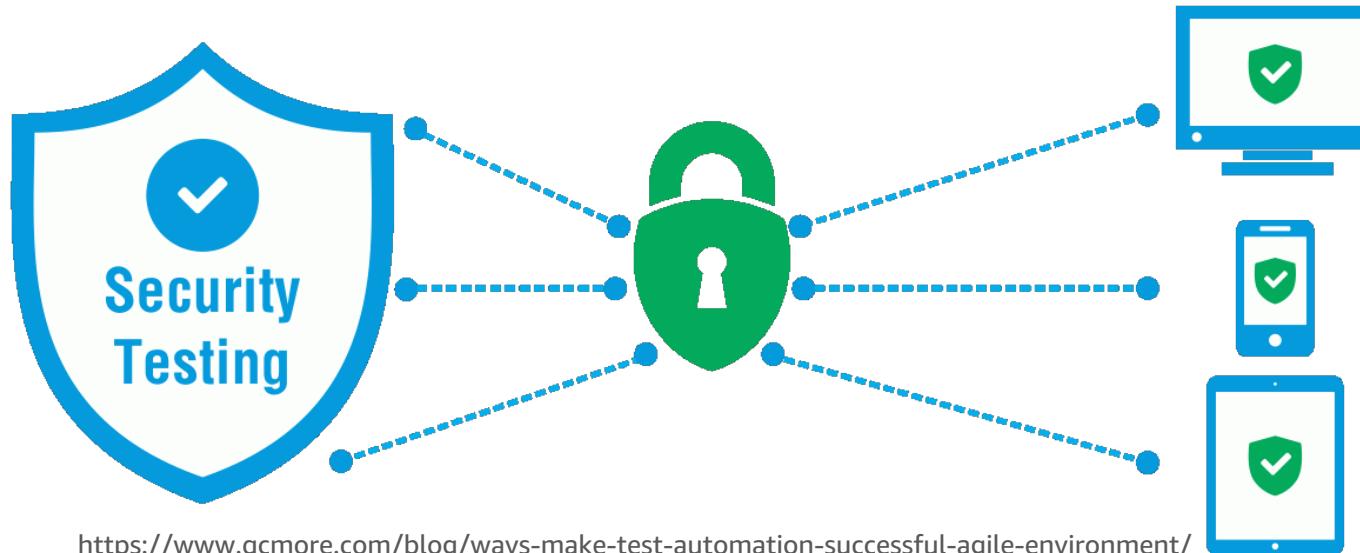
DevSecOps 아젠다



Test Driven Security (TDS)

Test Driven Security (TDS)

- 테스트 주도 보안 (TDS)
 - Continuous Security를 위한 첫번째 단계
 - 보안 컨트롤을 정의하고 구현 및 테스트
 - 보안을 제품의 기능으로 바라보는 관점
 - 모든 보안 테스트를 자동화



Test Driven Security (TDS)

- TDD와 TDS는 어떻게 다른가?!
 - 보안을 제품의 기능으로 다루는 점에서 유사함
 - 테스트의 대상은 Security Control

Test Security Control Like TDD

Test Driven Security (TDS)



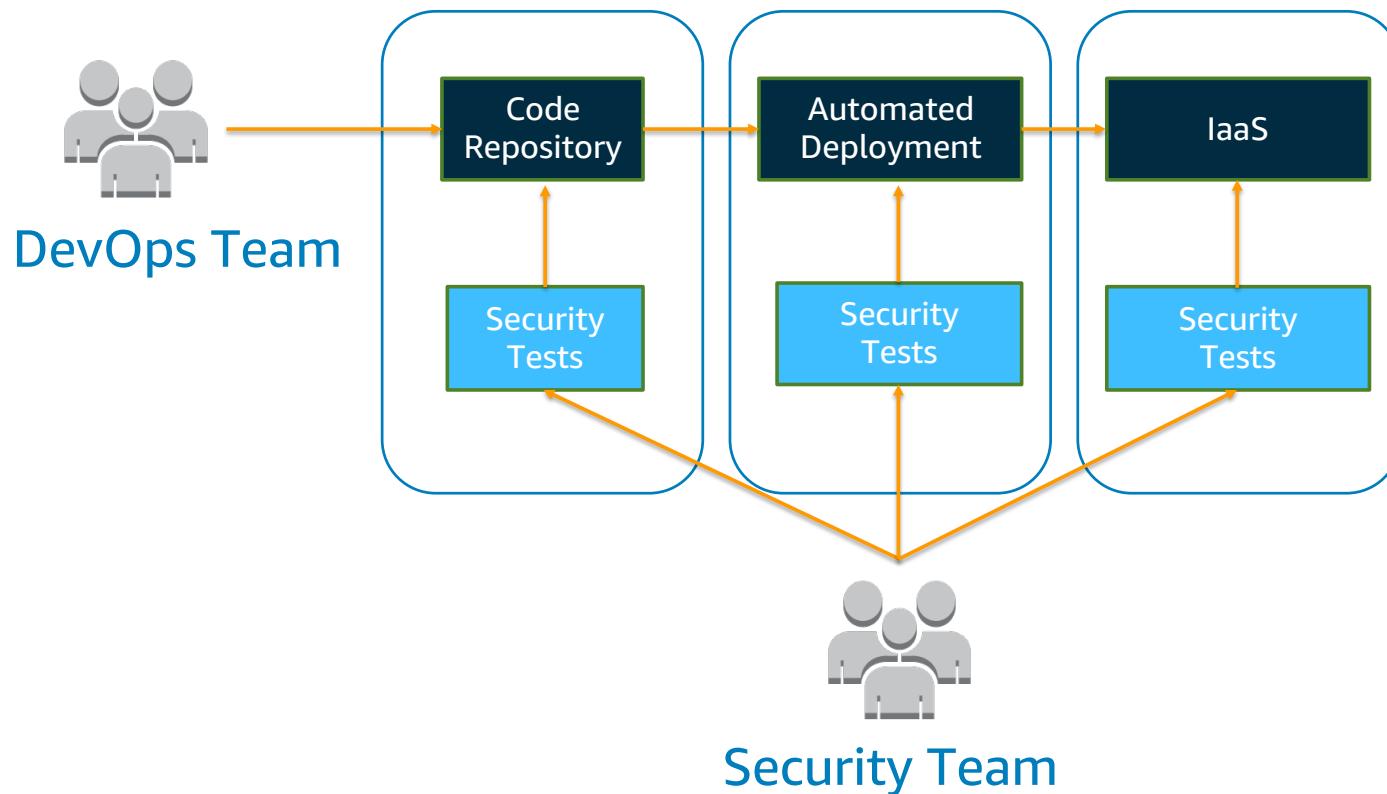
"TDS는 그 조직이 얼마나 안전한지에 대한 지속적인 그림을 보여주는 중요한 활동이다."

Justin Berman – Chief Information Security Officer of Zenefits

<https://www.youtube.com/watch?v=mBsNf3z34Sg&feature=youtu.be>

Test Driven Security (TDS)

- TDS는 파이프라인을 자동화하고 팀과 가깝게 협업하는 DevOps 원칙을 따른다



Test Driven Security (TDS)

- TDS 접근 방식의 이점
 - 테스트 작성을 통해 문제를 명확히 예상할 수 있고 필수 컨트롤에 대한 충분한 지식을 갖게 된다.
 - 컨트롤을 테스트하기 쉽고 작은 단위로 구성하여 모두가 이해할 수 있다.
 - 제품 및 서비스와 같은 인프라를 공유하므로 테스트의 재사용성이 높음.
 - 일단 일련의 기본 테스트가 작성되면 보안팀은 더욱 복잡한 작업에 집중할 수 있다.
 - 개발자와 운영자가 고객을 위험에 빠트리기 전에 문제를 해결할 기회를 제공

Test Driven Security (TDS)

- 전통적인 IT 환경에서의 TDS
 - TDS를 구현하기가 어려움
 - 수년간 동작중인 라이브 시스템에서 실행되어야 하는 부담감
 - 인프라에 대한 컨트롤이 어려움
- DevOps 환경의 TDS
 - DevOps에서는 소프트웨어 또는 인프라에 대한 모든 변경이 CI/CD 파이프라인을 통해 이루어짐
 - 최근의 Cloud Infra를 이용하여 Control에 대한 구현을 API과 SDK를 통해 구현 할 수 있음
 - 자동화된 파이프 라인을 통해 릴리즈 속도를 저하 시키지 않음

Test Driven Security (TDS)

어떻게 TDS를 구현 할 것인가?!

1. 베이스라인 정의하기
2. 테스트 작성하기
3. 베이스라인 테스트하기
4. 지속적인 테스트 실행
5. 배포시 테스트를 반드시 통과해야 함

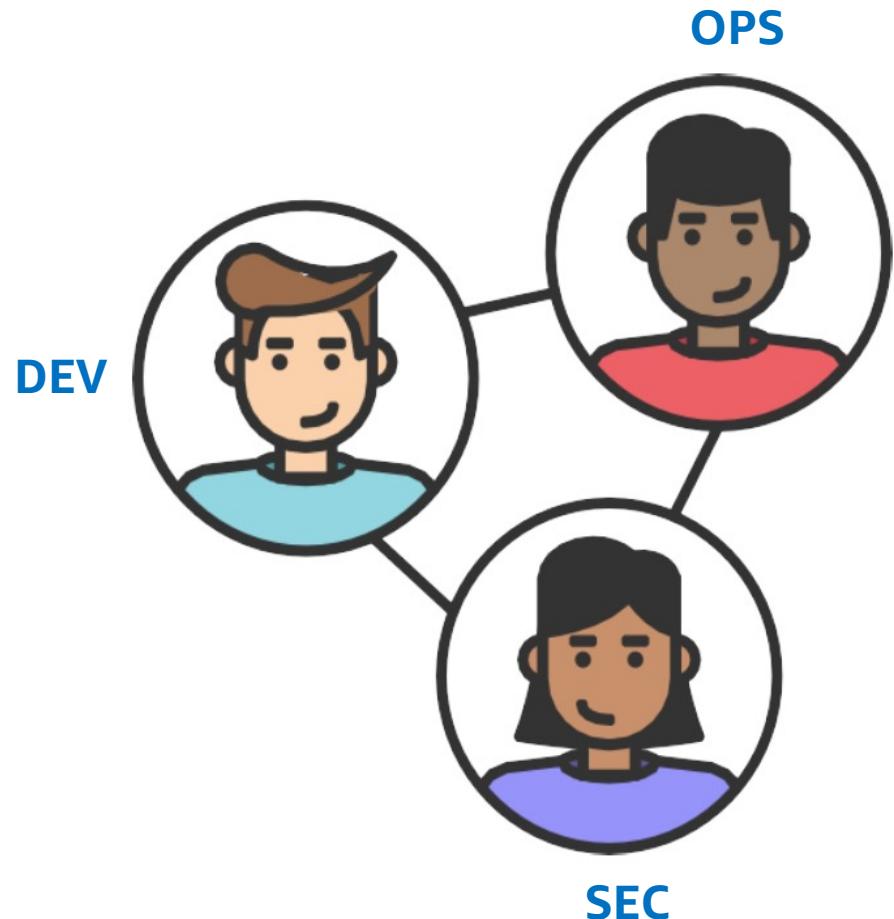


By Julien Vehent Firefox Operations Security Lead, Mozilla

<https://www.usenix.org/conference/enigma2017/conference-program/presentation/vehent>

Test Driven Security (TDS)

- 베이스라인 정의하기



“모든 사람이 구현하기를 원하는 보안 컨트롤 ”

“최소 보안 요구 사항의 집합 ”

“합의된 Top Priority”

Test Driven Security (TDS)

- 테스트 작성하기
- 예시: 컨테이너 이미지 보안 스캐닝 결과가 Critical이면 배포 파이프 라인을 중지 시킨다

```
...
code_pipeline = boto3.client('codepipeline')
ecr_client = boto3.client('ecr')
IMAGE_SCAN_CHECK_LEVEL = ['CRITICAL', 'HIGH']

...
for level in IMAGE_SCAN_CHECK_LEVEL:
    if level in scan_result:
        raise Exception(scan_result)
    # Get the list of artifacts passed to the function
    input_artifacts = job_data['inputArtifacts']
    put_job_success(job_id, 'message: ok')
```

Test Driven Security (TDS)

- 테스트하기 / 지속적인 테스트 실행
 - 코드가 커밋되기 전 로컬 테스트
 - CI서버에서 TDS 자동 실행
 - AWS 에서는 CodeBuild 혹은, CodePipeline을 테스트 러너로 사용 가능

Test Driven Security (TDS)

- TDS의 테스트가 실패하면?
 - 파이프라인의 실행은 중단됨
 - TDS 성공하는 조건을 만족 시킬 때 까지..



TRY AGAIN!

TDS를 위한 Security Baseline 을 생각해 봅시다. 5min



Test Driven Security (TDS)

- TDS 의 베이스라인 예시

- SSH 루트 로그인은 모든 시스템에서 비활성화해야 한다.
- 시스템과 애플리케이션은 출시 후 30일 이내에 최신 버전으로 패치
- 웹 애플리케이션은 HTTP가 아닌 HTTPS를 사용해야 한다.
- 자격증명은 애플리케이션 코드와 함께 저장될 수 없다.
- 관리자 화면은 VPN을 통해서만 접근할 수 있다.



[This Photo](#) by Unknown Author is licensed under [CC BY-SA](#)

Test Driven Security (TDS)

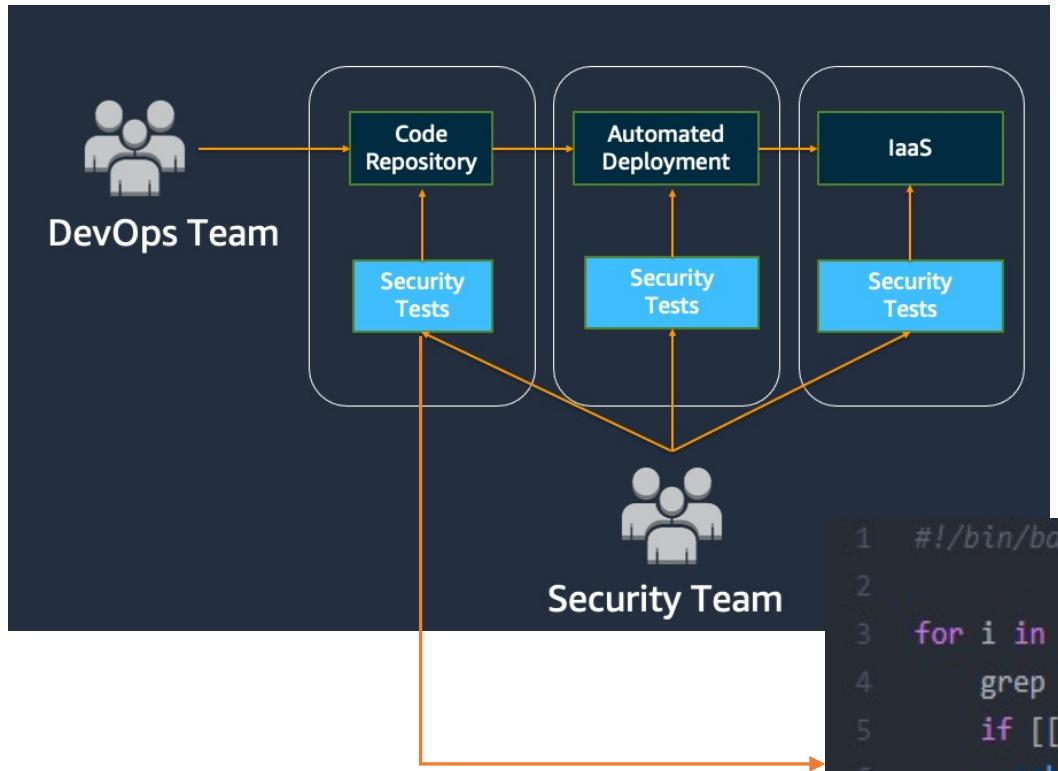
- 베이스 라인 예시
 - 인프라스트럭처
 - 로그 아카이브는 최소한 90일치가 보관되어야 한다.
 - 시스템간의 통신은 TLS를 통해서 한다.
 - 배포
 - 모든 릴리즈 태그와 커밋은 추적성 확보를 위해 사인을 한다.
 - Third Party 라이브러리는 whitelist로 관리한다.
 - ...

<https://www.usenix.org/conference/enigma2017/conference-program/presentation/vehent>

© 2021, Amazon Web Services, Inc. or its Affiliates. All rights reserved.



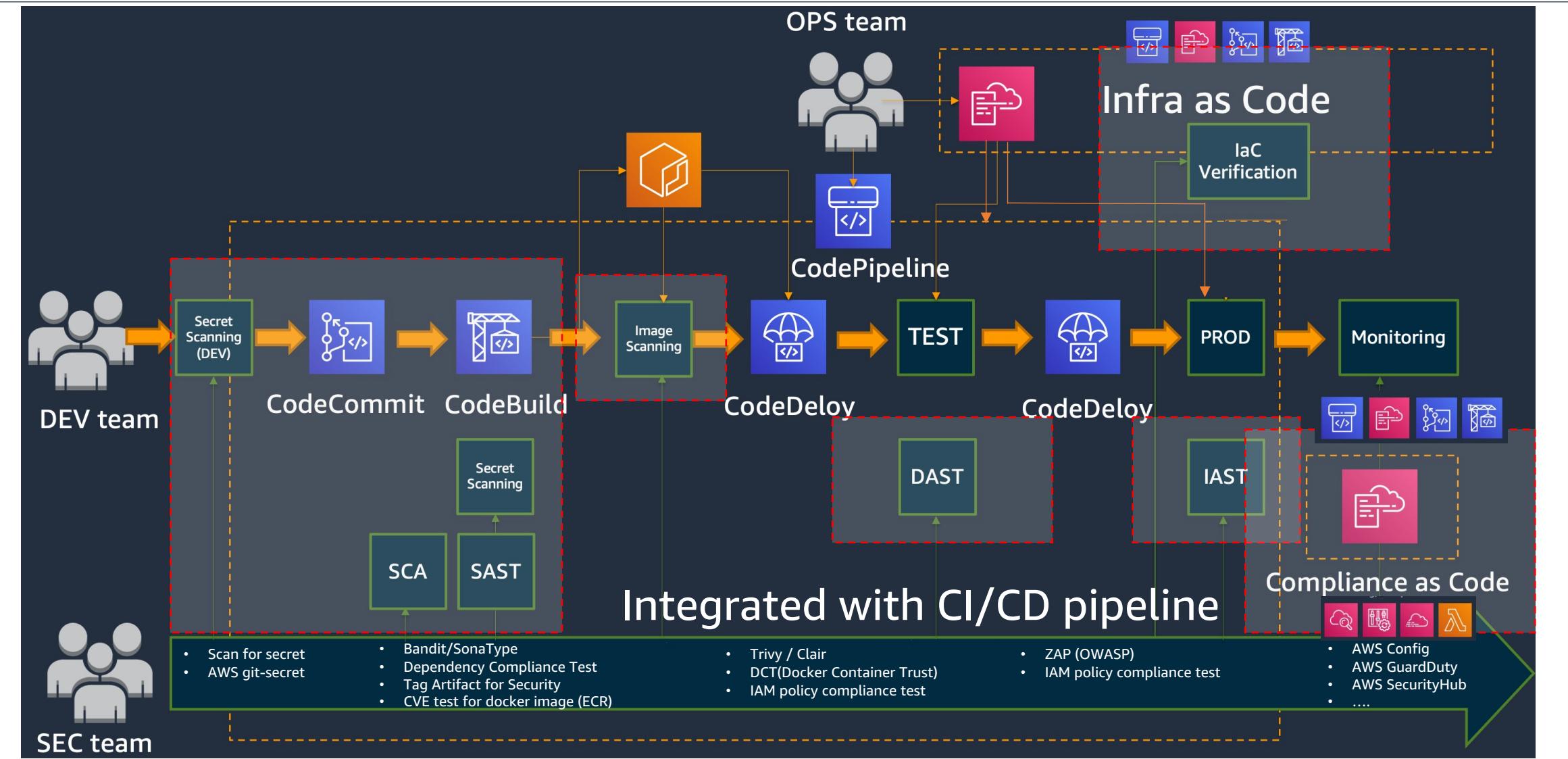
Test Driven Security (TDS)



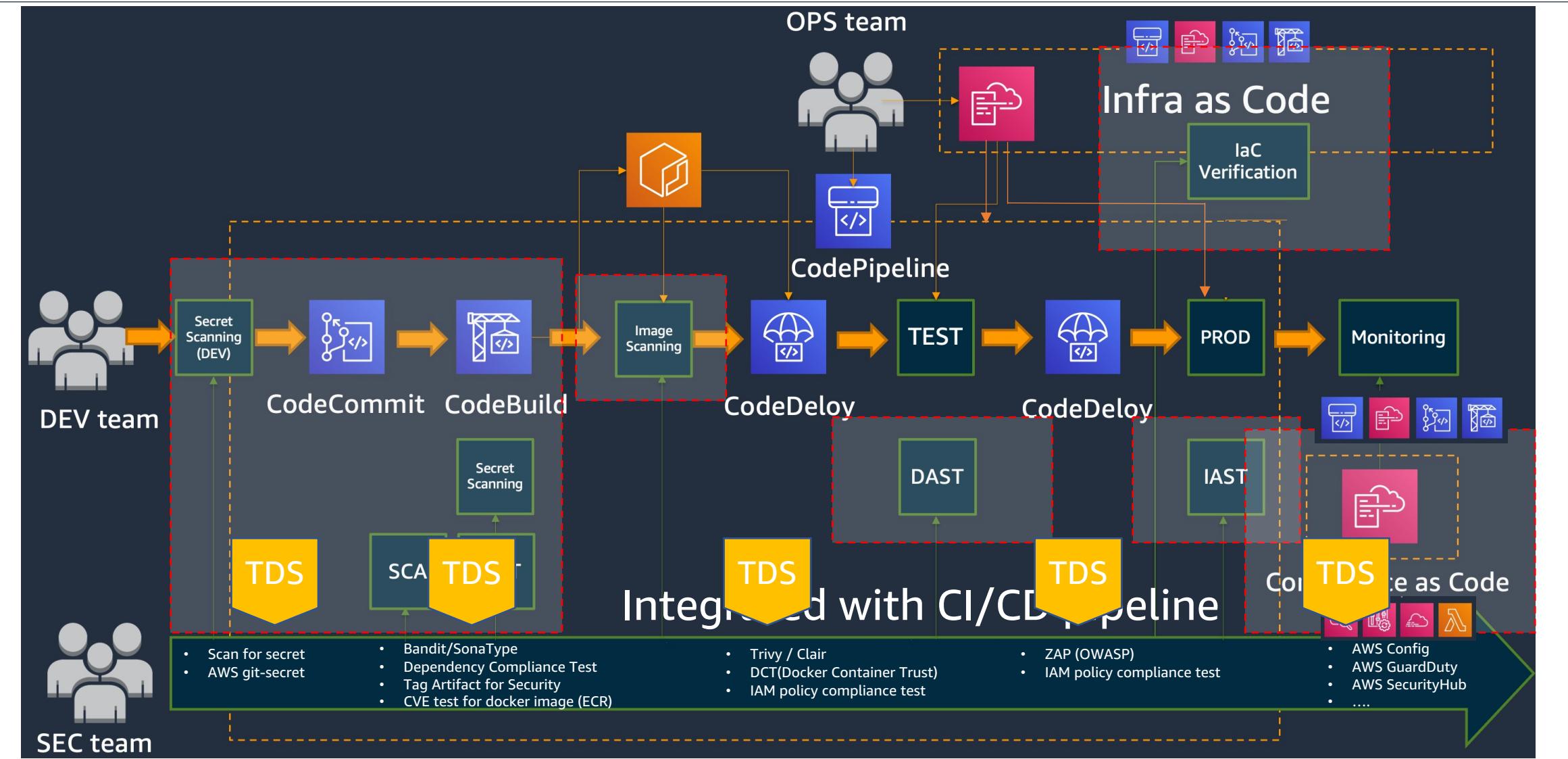
Sample security test script

```
1 #!/bin/bash
2
3 for i in `cat requirements.txt`; do
4     grep -q $i whitelist.txt
5     if [[ $? == 1]]; then
6         echo "Security test failed: unauthorized package in requirements.txt"
7         exit 1
8     fi
9 done
10
```

Security Testing



Security Testing



Compliance as Code

Compliance as Code

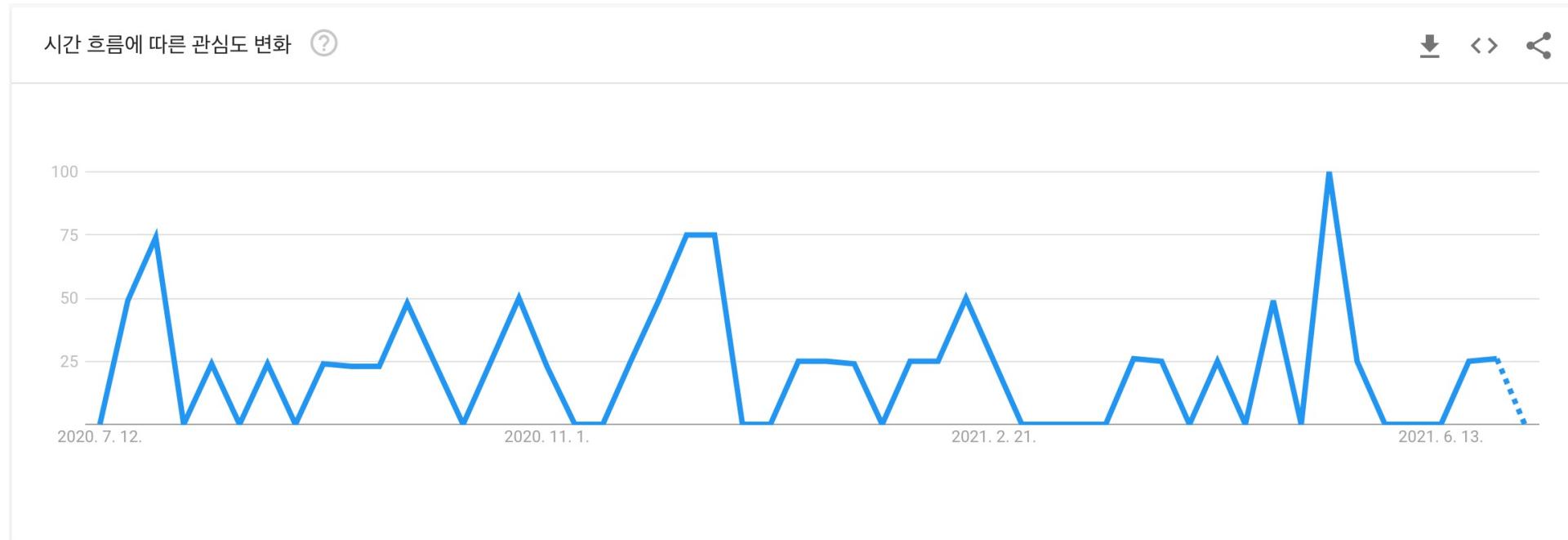
- 컴플라이언스 관리체계를 코드 기반의 파이프라인으로 이동하는 아이디어
- 코드는 클라우드에서 실행되며 컴플라이언스 검증에 필요한 리소스를 지속적으로 평가
- 규정준수 가시성 확보



<https://amazicworld.com/what-compliance-as-code-means-for-your-business/>

Compliance as Code

- Infra as code



from 2020.07 ~ to 2021.07

Compliance as Code

- Infra as code vs Compliance as code



- Compliance as code

Compliance

분야	항목	상세내용	주요 확인사항	비고	
	2.2.5	퇴직 및 직무변경 관리	퇴직 및 직무변경 시 인사·정보보호·개인정보보호·IT 등 부서에 따른 정부기관 및 자산반납·계정 및 접근권한 회수·조정·결과확인 등의 절차를 수립·관리하여야 한다.	조직 내 인력(임직원, 임시직원, 외주용역직원 등)이 퇴직 또는 직무변경 시 인사·정보보호·IT 등 부서에 따른 정부기관 및 접근권한 회수·조정·결과확인 등의 절차를 수립·관리하여야 한다.	▣ 관련 기록을 보존하고 퇴직 절차 준수여부에 대하여 정기적으로 점검 ▣ 개인정보처리시스템에 대한 점검권한을 가지고 있는 개인정보처리자의 경우 개인정보보호법 최소 3년, 정보통신망법 최소 5년 보관
	2.2.6	보안 위반 시 조치	임직원 및 관련 외부자가 법령·규제 및 내부정책을 위반한 경우 이에 따라 조치 절차를 수립·이행하여야 한다.	임직원 및 관련 외부자가 법령과 규제 및 내부정책에 따른 정보보호 및 개인정보보호 책임과 의무를 위반한 경우에 대한 처벌 규정을 수립하고 있는가?	▣ 법규 위반 발견 시 조사, 소명, 징계 등 조치 기준 및 철저 수립 하고 있어야 하는 경우 보상 방안도 고려 적용
2.3. 외부자 보안	2.3.1	외부자 현황 관리	업무의 일부(개인정보취급, 정보보호 및 정보시스템 운영 또는 개발 등을)를 외부에 위탁하거나 외부 서비스(인터넷 및 모바일 서비스·인터넷방송·온라인 쇼핑·클라우드 서비스·애플리케이션 서비스 등)를 이용하는 경우 그 현황을 적발하고 개인정보 보호 대책을 마련하여 직접 서비스로부터 발생되는 위험을 파악하고 적절한 보호대책을 마련하여야 한다.	관례체계 별의 내에서 발생하고 있는 업무 위탁 외부 시설·서비스의 이용 현황을 식별하고 있는가?	
	2.3.2	외부자 계약 시 보안	외부 서비스를 이용하거나 외부자에게 업무를 위탁하는 경우 이에 따른 정보보호 및 개인정보보호 요구사항을 식별하고 이를 계약서 제작이나 또는 업무상 통해 당시 마련 하였어야 한다.	종료일 및 개인정보 처리와 이용 목적 외부 서비스 제공 및 업체를 설정하는 경우 정보보호 및 개인정보 보호 대책을 마련하도록 체결을 마련하고 있는가?	
	2.3.3	외부자 보안 이행 관리	계약서·협정서·내부정책에 명시된 정보보호 및 개인정보보호 요구사항에 따라 외부자의 보호대책 이행 여부를 주기적으 경검 또는 감사 등 관리·감독하여야 한다.	외부 서비스 이용 및 업무 위탁에 따른 정보보호 및 개인정보보호 요구사항을 식별하고 이를 계약서 또는 협정서에 명시하고 있는가?	
			개인정보처리시스템 개발을 위탁하는 경우 개발 시 준수해야 할 정보보호 및 개인정보보호 요구사항을 계약서에 명시하고 있는가?	개인정보처리시스템 개발을 위탁하는 경우 개발 시 준수해야 할 정보보호 및 개인정보보호 요구사항을 계약서에 명시하고 있는가?	

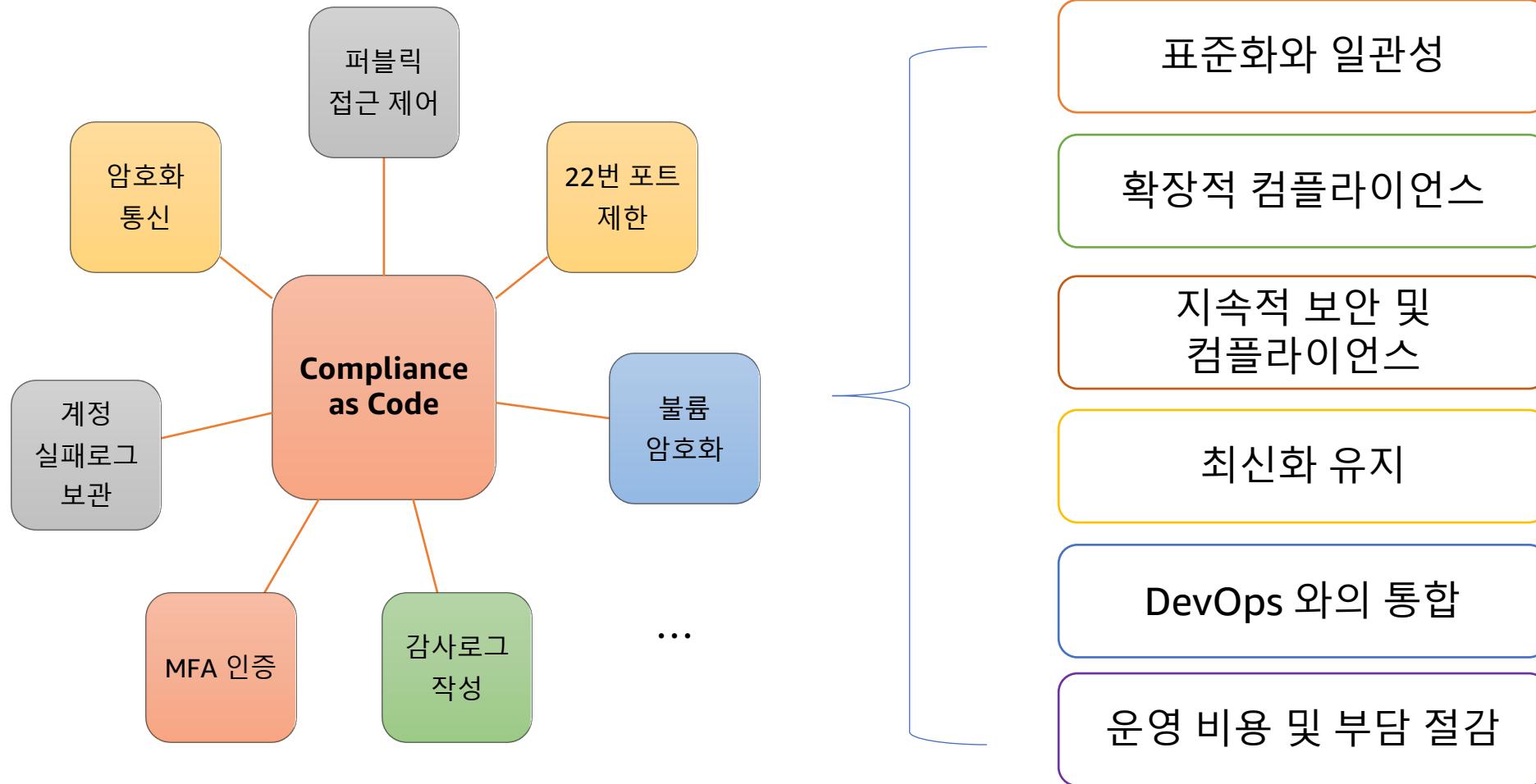
임직원 권한 부여 관련 life cyce 관리

30일 이상 미사용 계정 점검

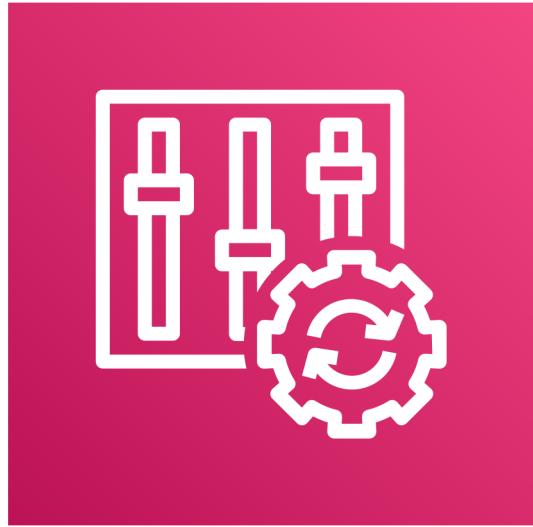
외부 인력 권한 부여 제한

수많은 제3기관 보안 요건들,
그리고 회사 내부 컴플라이언스..

Compliance as Code 의 장점



Compliance as Code on AWS



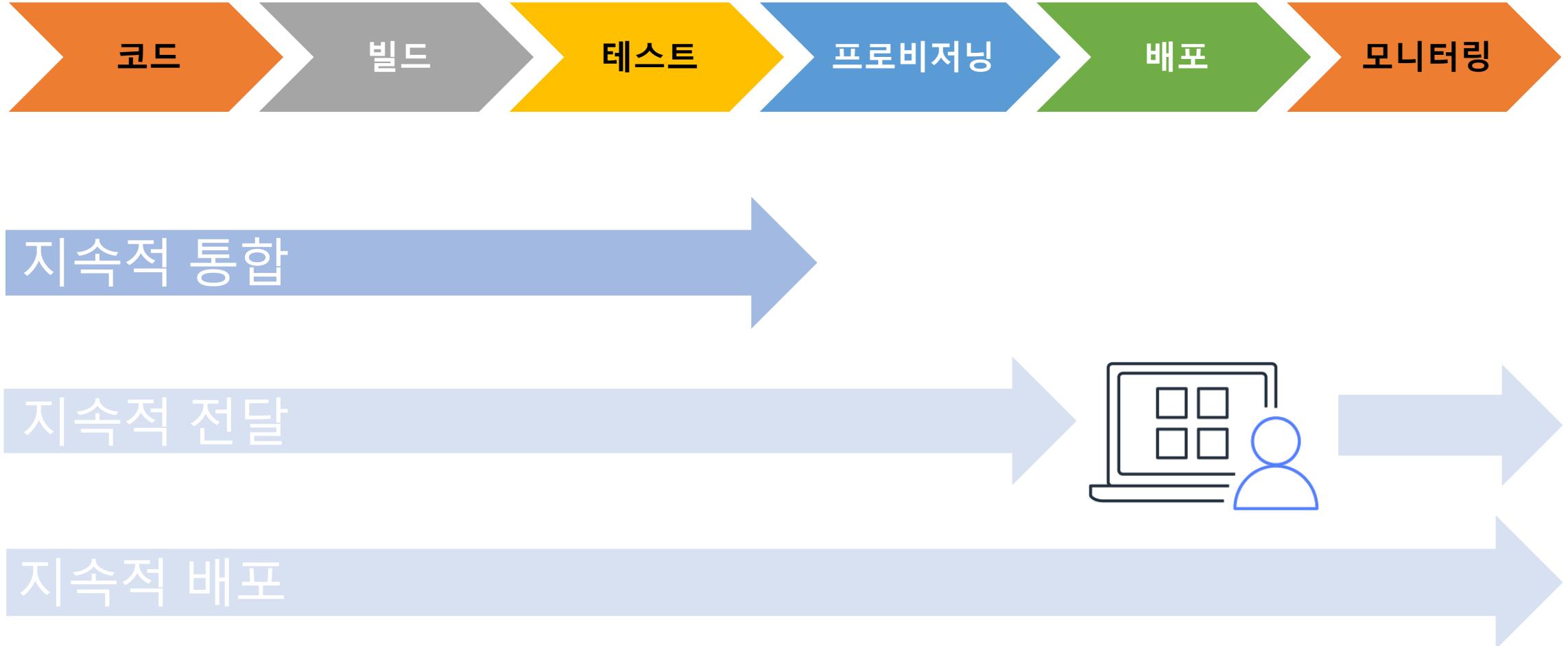
AWS Config



AWS CloudFormation
(Guard)

CICD Pipeline Security

CI/CD 파이프라인



DevSecOps CI/CD 파이프라인

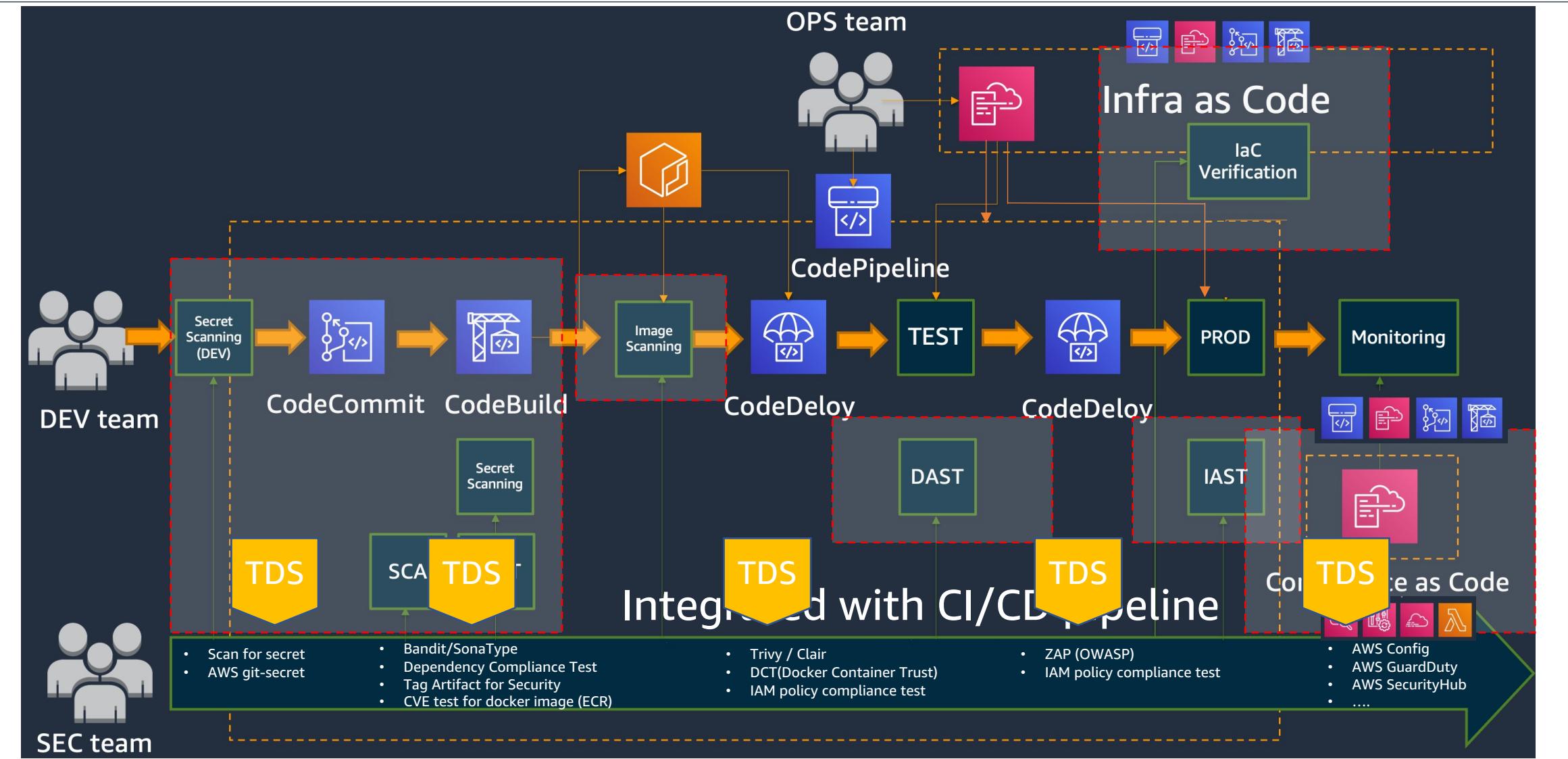
1. CI/CD 파이프라인의 보안

- 액세스 역할
- 빌드 서버/노드 강화

2. CI/CD 파이프라인에서의 보안

- 아티팩트 검증
- 정적 코드 분석

Security Testing

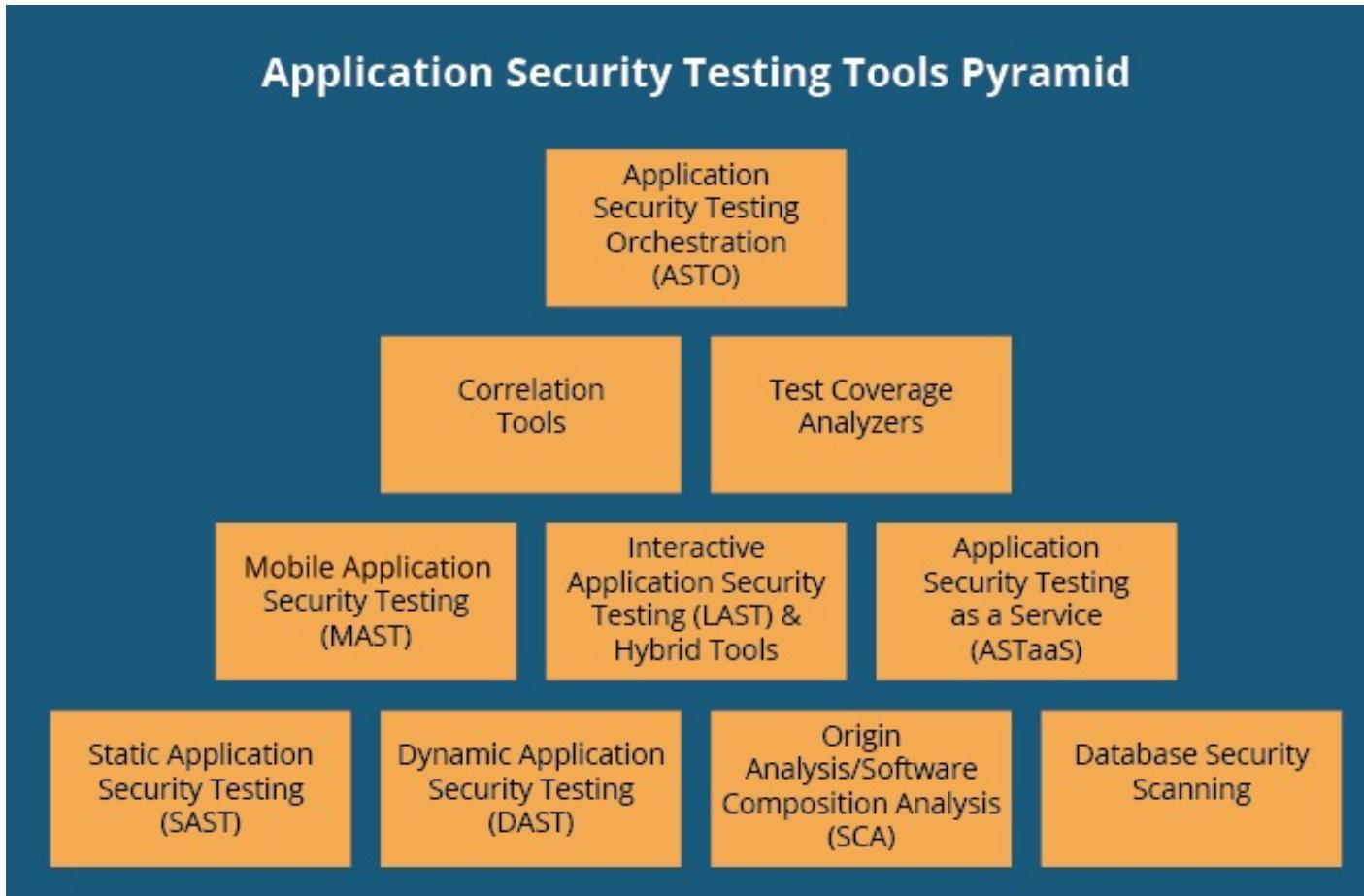




어떤 테스트를 하고 계십니까?

Security Testing

- Application Security Testing 도구 피라미드

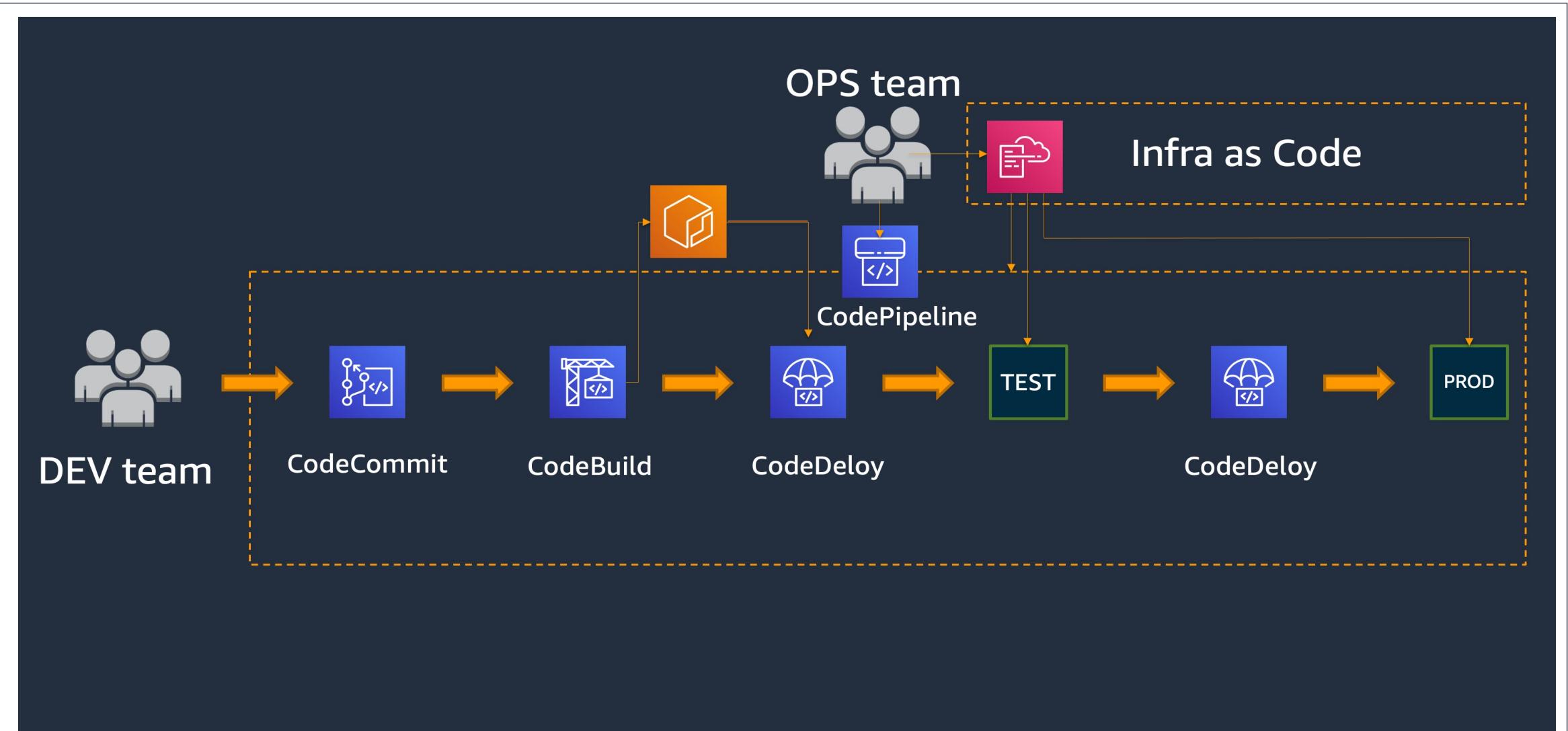


<https://insights.sei.cmu.edu/blog/10-types-of-application-security-testing-tools-when-and-how-to-use-them/>

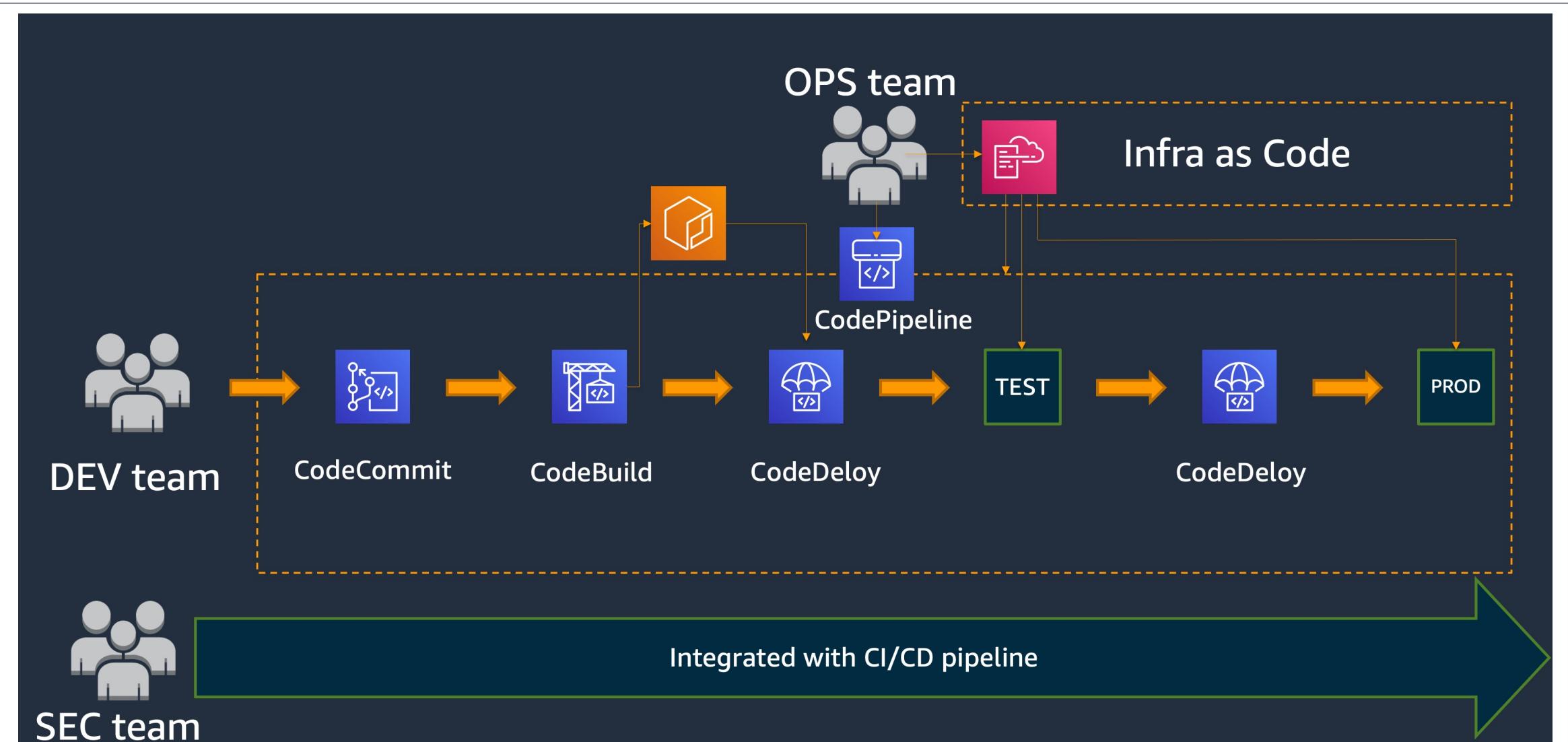
Security Testing

- AppSec TEST
 - **SAST**: Static Application Security Testing
 - **SCA**: Software Composition Analysis
 - **DAST**: Dynamic Application Security Testing
 - **IASA**: Interactive Application Security Testing

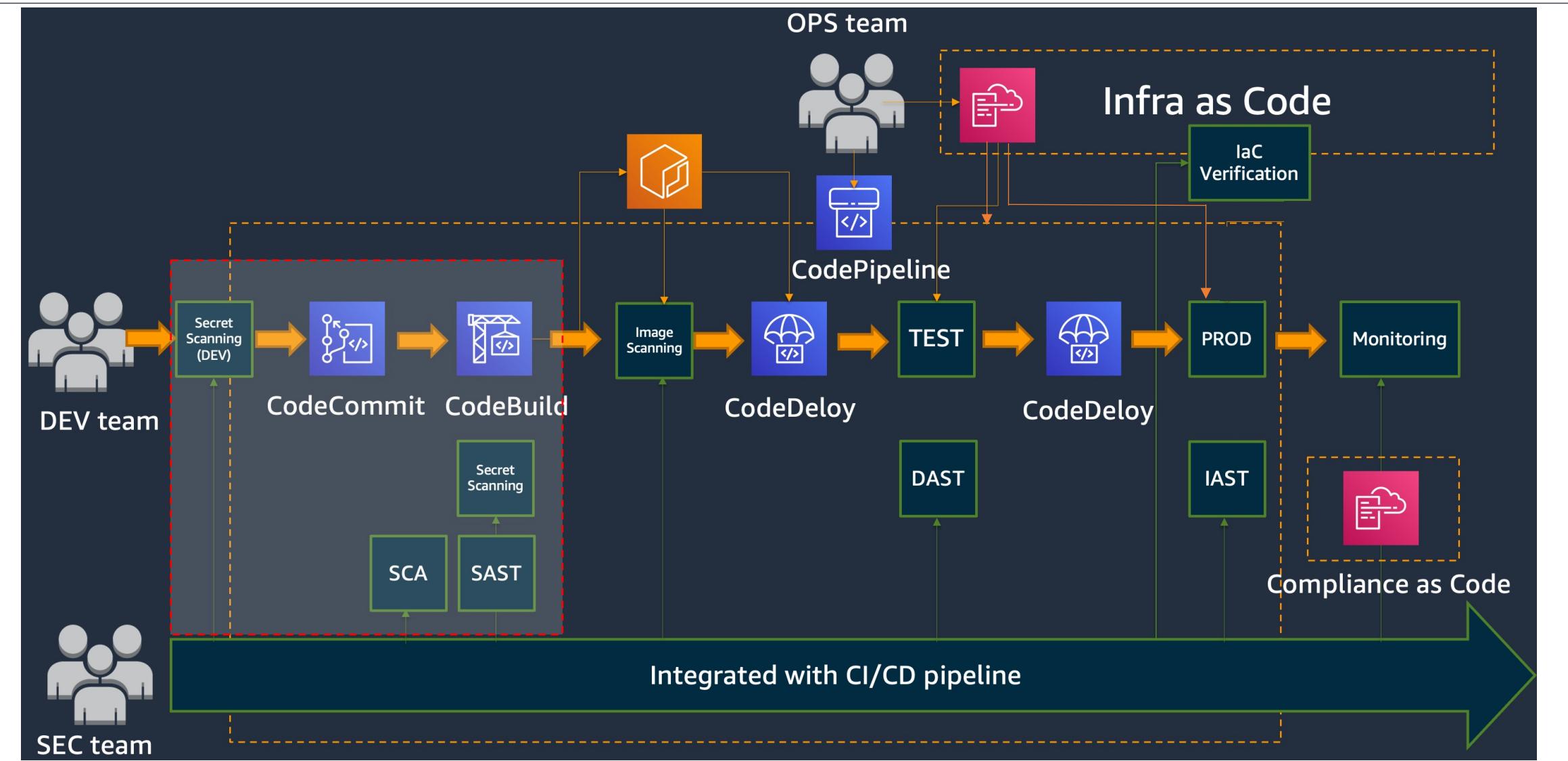
Example of CI/CD pipeline



Example of CI/CD pipeline



Security Testing



Security Testing

- AST 진행시 고려 할 점
 - Integration
 - 얼마나 쉽게 CI/CD에 통합 할 수 있는가?
 - Accuracy
 - False positives 는 어느 정도인가?
 - Speed
 - 테스트를 실행하는데 어느정도 빠른가?
 - Actionability
 - 실행 가능성이 충분한가? 명확한 지침이 있는가?
 - 노이즈 시그날이 있지 않은가?

Security Testing

- Static Testing (aka SAST)
 - 잠재적 보안 문제 혹은 안전하지 않은 코드 스캔
 - Challenges
 - 오탐과 노이즈에 대한 튜닝 필요
 - Exploitable 이슈와 non-exploitable 이슈를 정확하게 구분하기 어려움
 - 런타임 컨텍스트를 가지고 있지 않은 상황에서의 테스트
 - Deployment
 - 개발자의 머신
 - IDE plugins (pre-commit)
 - CI의 한 부분
 - Commit된 후 코드 통합 단계

Security Testing

- Static Testing (aka SAST) - 장점 및 단점
 - Integration
 - 쉬움
 - Accuracy
 - False-Positive에 대한 고려 필요
 - Speed
 - 수분 혹은 수 시간
 - Actionability
 - 코드 라인별로 결과를 볼 수 있지만 real-issue 인지는 파악이 필요

Security Testing

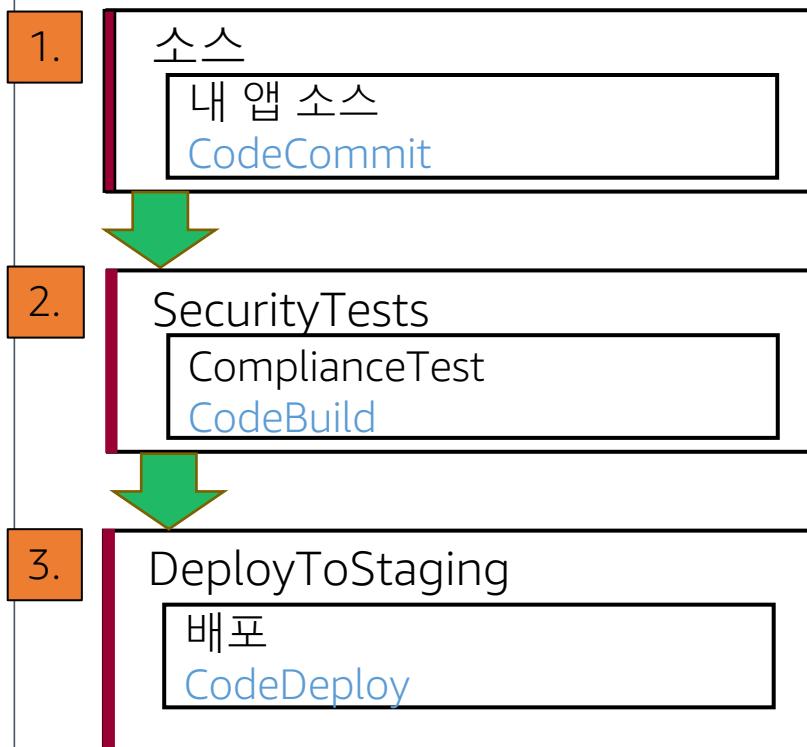
- Static Testing (aka SAST) - 도구들
 - 상용
 - Checkmarx
 - Microfocus
 - Synopsys
 - etc
 - OSS
 - Java (Sonarqube), Python(Bandit), Ruby(Brakeman, Cane), PHP(PHPStan)..
 - <https://github.com/mre/awesome-static-analysis>
 - IDE 통합
 - 대부분의 상용 도구에서 제공

Security Testing

- Software Composition Analysis (SCA)
 - 써드파티 디펜던시에 대한 Risk 를 줄이기 위한 테스트
 - OSS 의존성 트리를 사용하여 취약점 점검(CVEs)
 - Tools
 - Snyk
 - Github Security Alerts
 - SourceClear

Security Testing

- Software Composition Analysis (SCA)



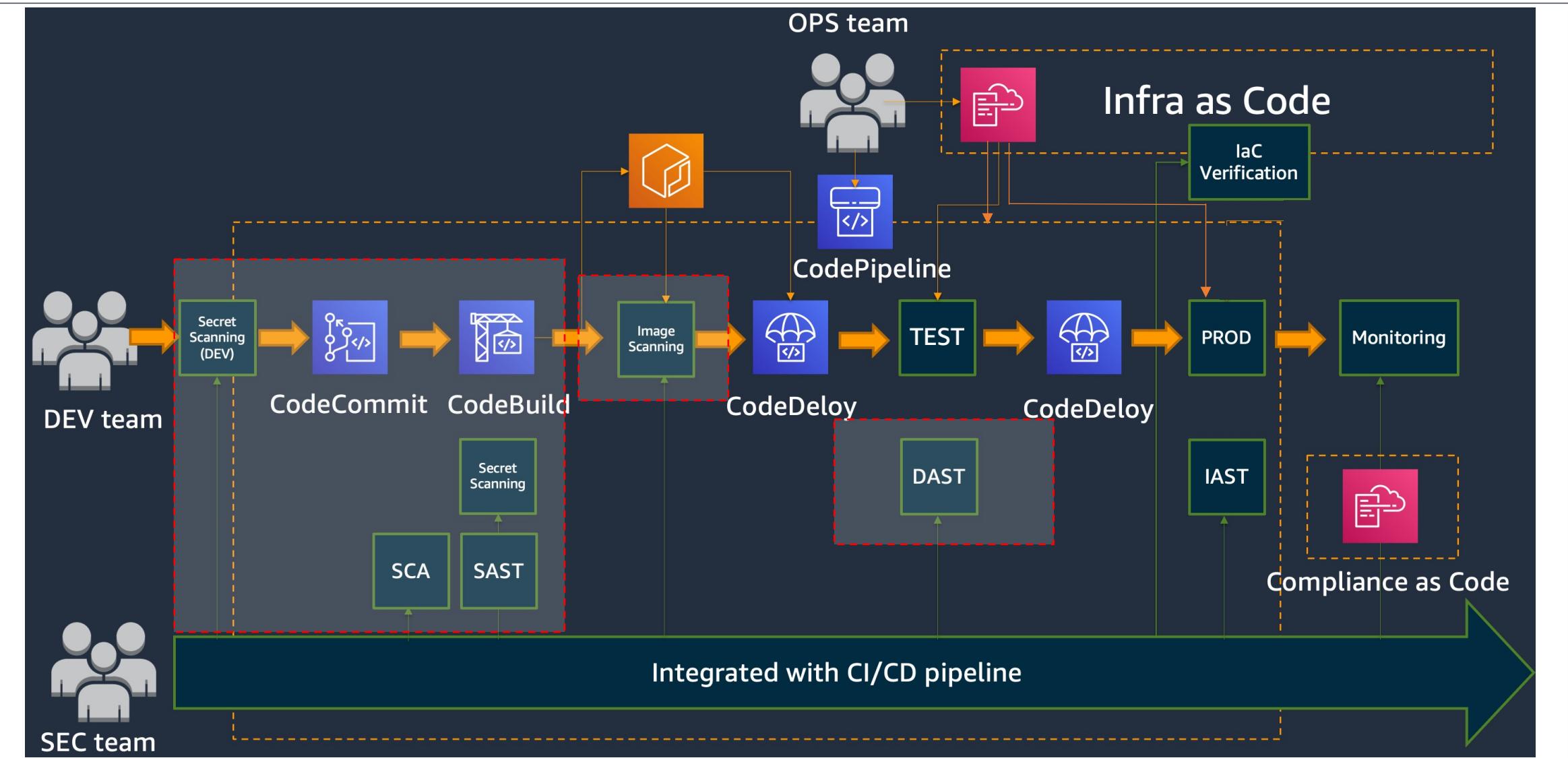
간단한 테스트 스크립트 예제

```
1  #!/bin/bash
2
3  for i in `cat requirements.txt`; do
4      grep -q $i whitelist.txt
5      if [[ $? == 1 ]]; then
6          echo "Security test failed: unauthorized package in requirements.txt"
7          exit 1
8      fi
9  done
10
```

Security Testing

- Secret detection
 - 개발자에 의해 하드코딩 시크릿 정보 탐지
 - API keys
 - AWS keys
 - oAuth Client Secrets
 - SSH Private Keys..
 - 도구들
 - git-secrets from awslabs (<https://github.com/awslabs/git-secrets>)
 - Tool for Yelp

Security Testing



Security Testing

- Dynamic Testing (aka DAST)
 - 애플리케이션에 HTTP 요청 기반 테스트
 - 라이브러리로 구축된 다양한 Payload 사용 가능
 - SQL injections, XSS, etc..
 - Fuzzing
 - 장점
 - 취약점 발견(real exploitable)
 - 런타임 컨텍스트에서 실제로 테스트 가능 (애플리케이션이 DB에 연결 된 채로 테스트됨)
 - Challenges
 - SAST보다 더 많은 시간이 걸림
 - 많은 경우 SPA에 대한 테스트가 어려움(API Scan이 잘 안됨, Wallarm FAST 가능)
 - 많은 경우 DAST는 CICD에 통합 시키기가 어려움

Security Testing

- Dynamic Testing (aka DAST) – CI/CD를 위한 체크사항
 - 다수의 툴들이 이미 개발이 완료되어 있음
 - Manual하게 실행되어야 하는 경우
 - Old fashioned apps에 맞춰져 있는 경우(SPA 테스트 어려움)
 - Requirements
 - 내가 사용하고 있는 CI와 통합이 되는가?
 - APIs에 대한 테스트가 가능한가? (SPA의 경우는 어떤지?)
 - DevOps 파이프라인의 속도를 저해 하지는 않는가?

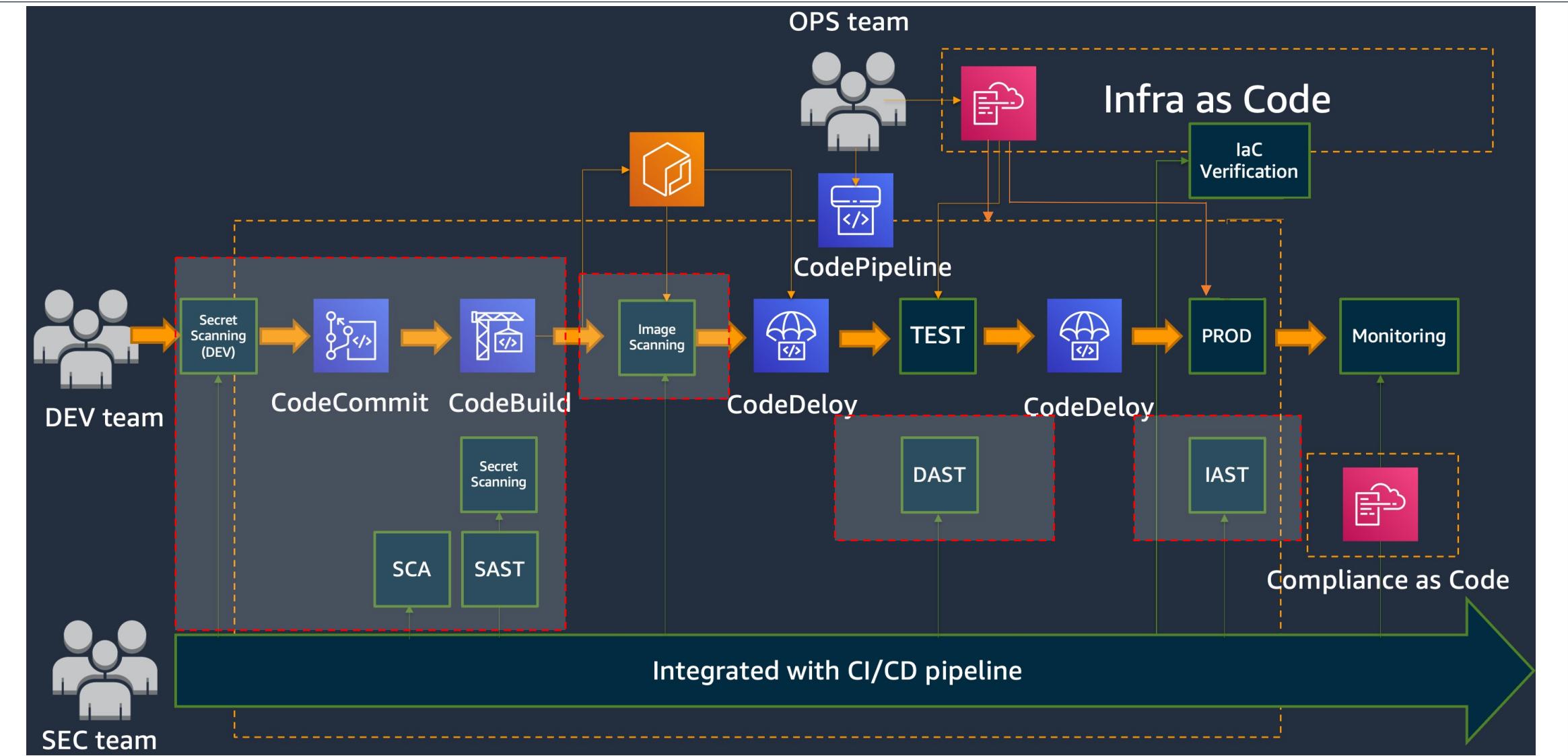
Security Testing

- Dynamic Testing (aka DAST) – 도구들
 - OWASP Zap(OSS)
 - Integration: Console, Command Line, API
 - API Testing : Challenging
 - Burp Enterprise (Commercial)
 - Integration: API
 - API Testing : Challenging
 - Wallarm Fast(Commercial) – DAST + Fuzzing
 - Integration: API
 - API : Strong

Security Testing

- Dynamic Testing (aka DAST) – 장점 및 단점
 - Integration
 - Test Automation
 - Accuracy
 - 정확도가 높고, 컨피규레이션이 단순함
 - Speed
 - 상황에 따라 시간이 오래 걸림
 - Actionability
 - 발견된 취약점은 현실 반영(정말로 위험함)
 - 특정 코드를 pinpoint 해줄 필요가 있음

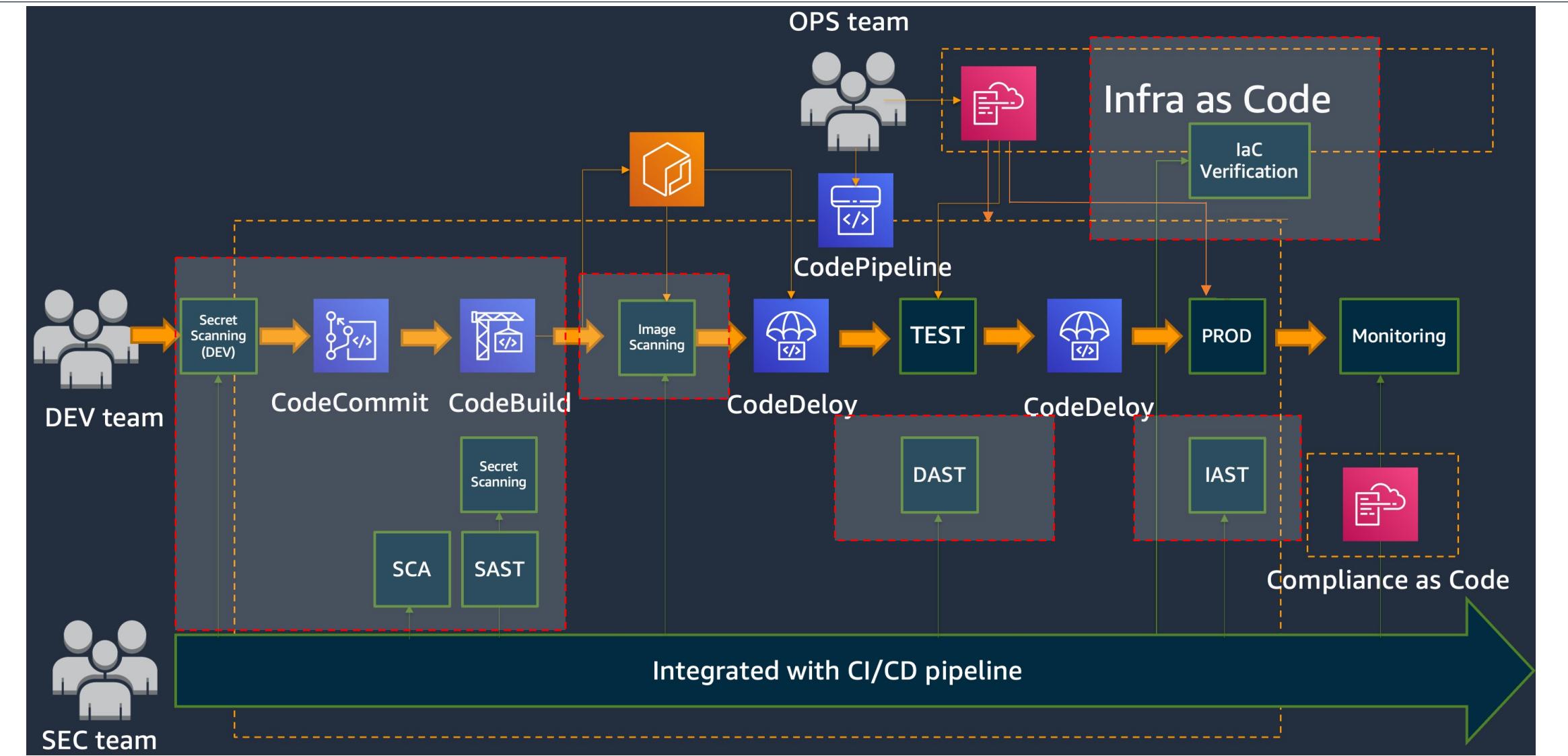
Security Testing



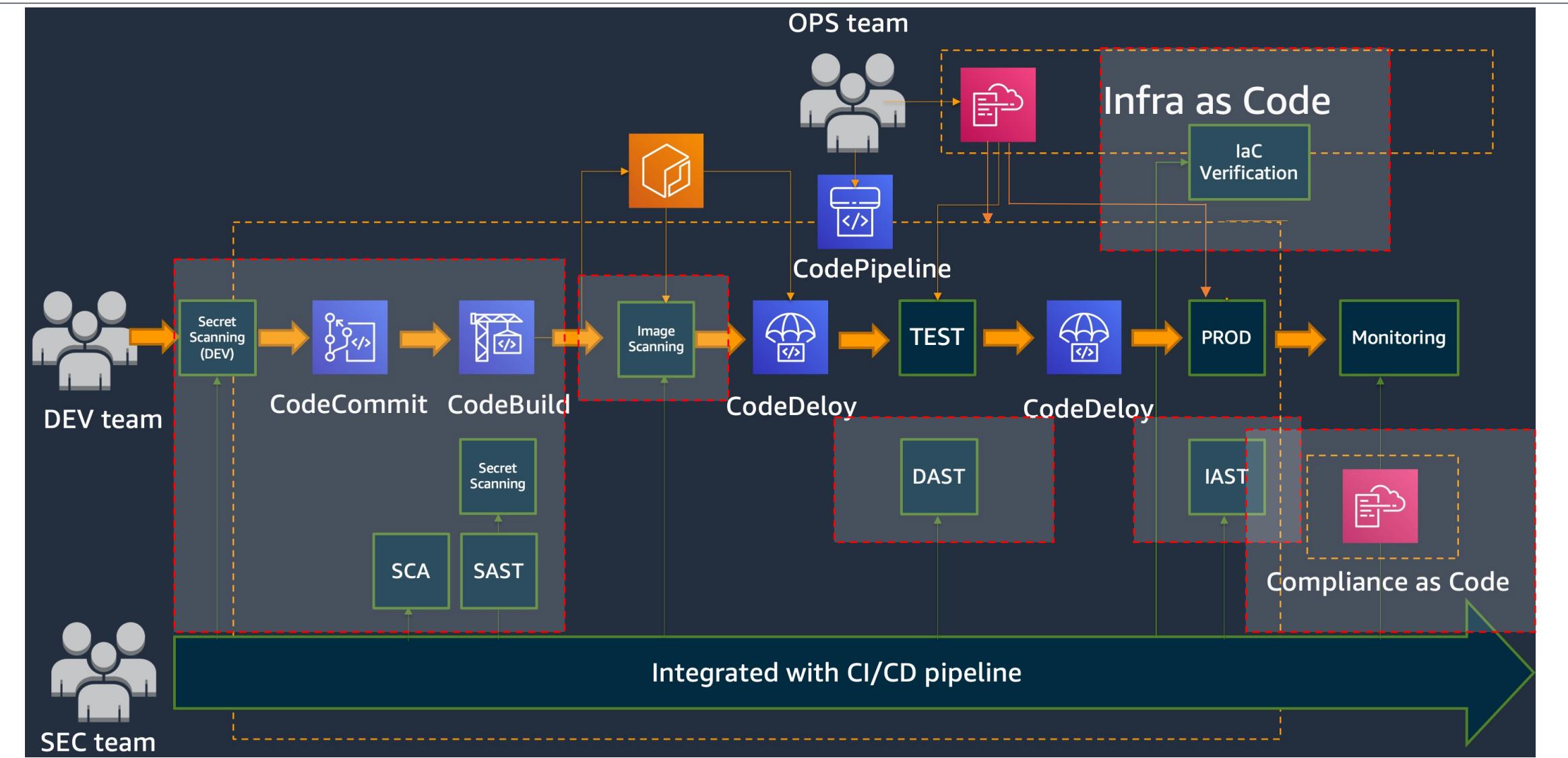
Security Testing

- Interactive Application Security Testing (IAST)
 - 스테이징/프로덕션 환경에서 사용 가능
 - SAST/DAST와 상호 보완적
 - 런타임환경에서 테스트하며 실해중인 코드 트래킹
- 도구들
 - Synopsys Seeker
 - Contrast Security Assets
 - Hdiv

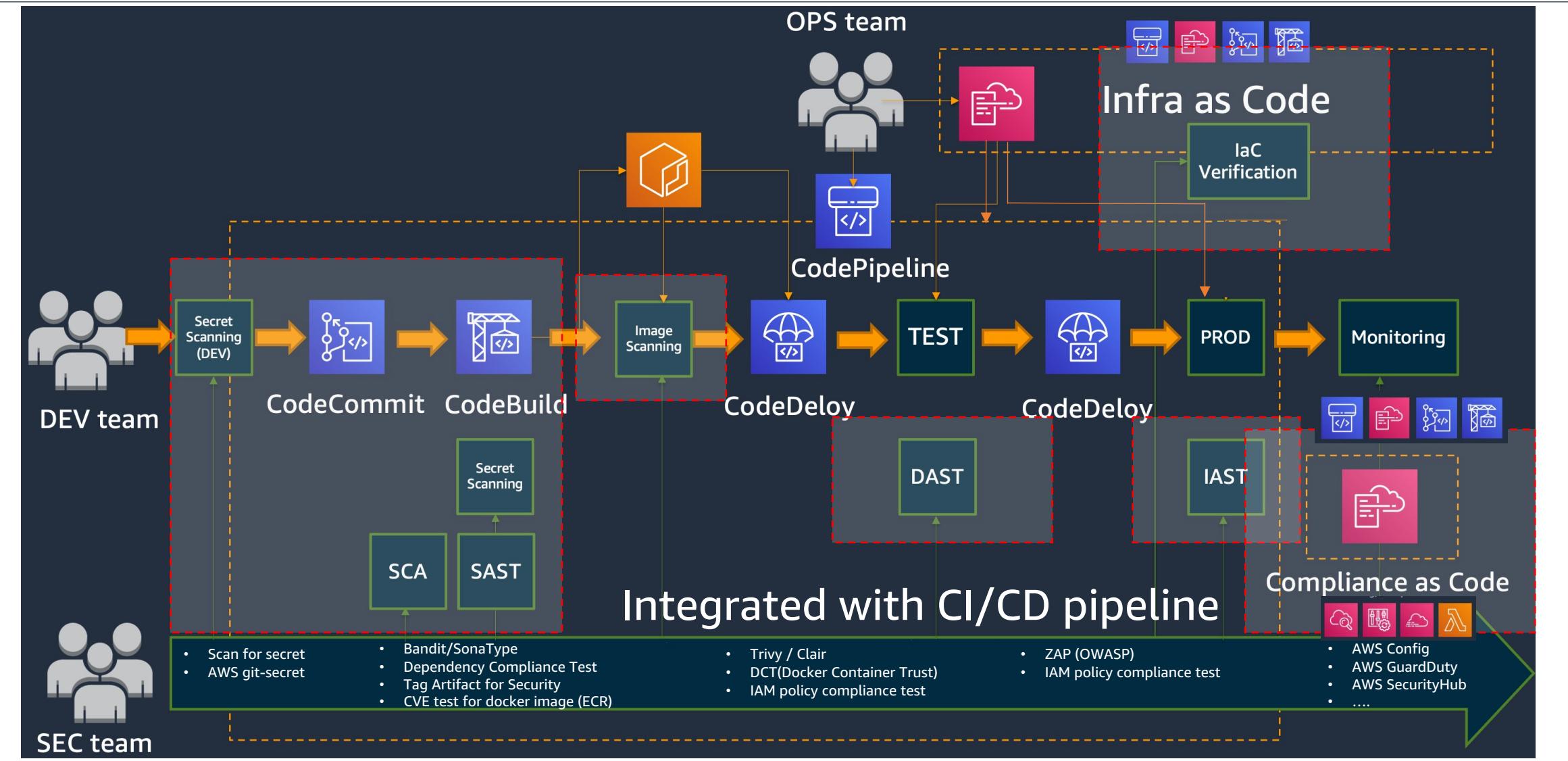
Security Testing



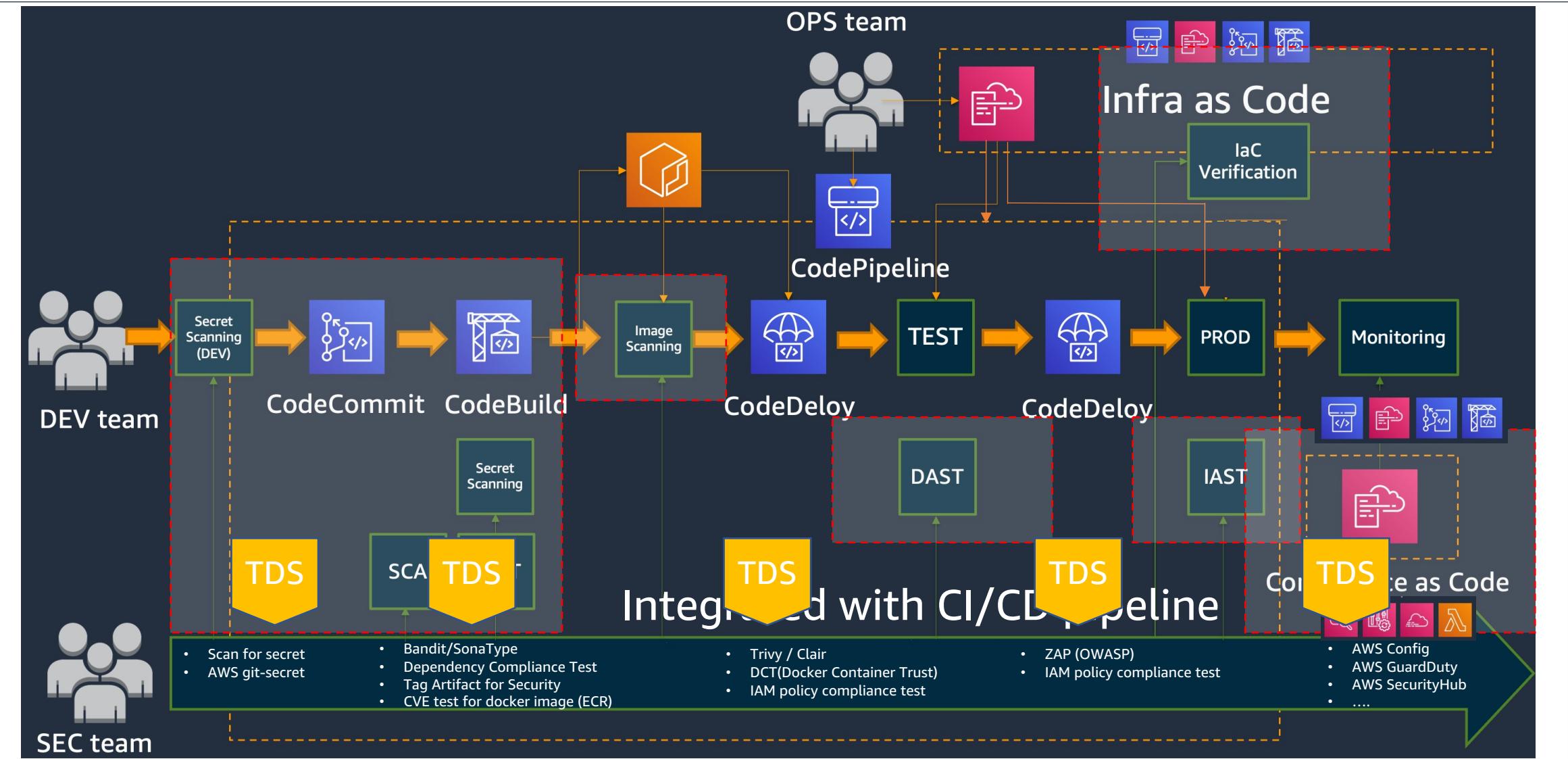
Security Testing



Security Testing

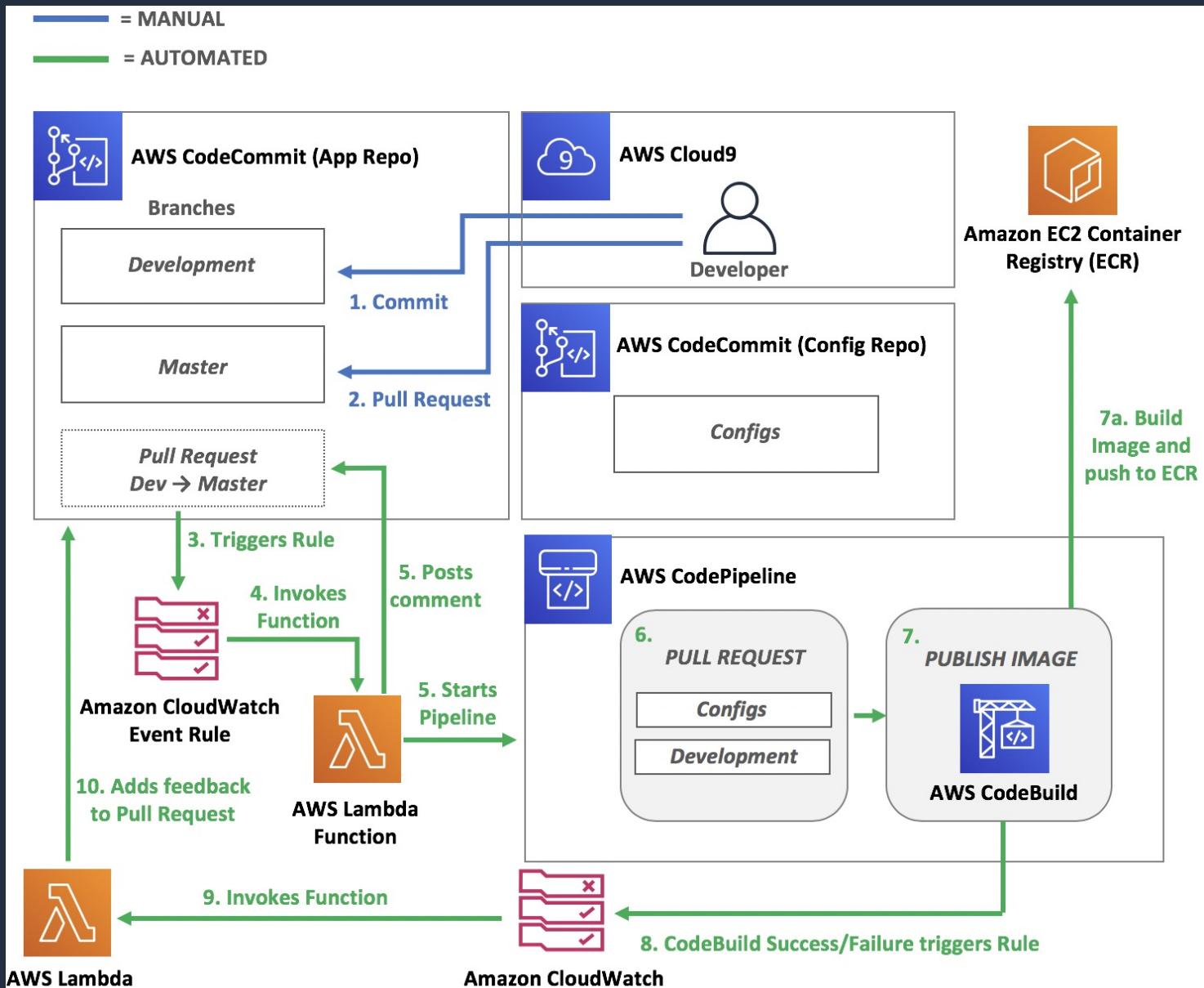


Security Testing





DEMO



- Hands-on Lab
 - <https://bit.ly/awskrug-tds> (한글)
 - <https://container-devsecops.awssecworkshops.com/> (영문)

Thank You