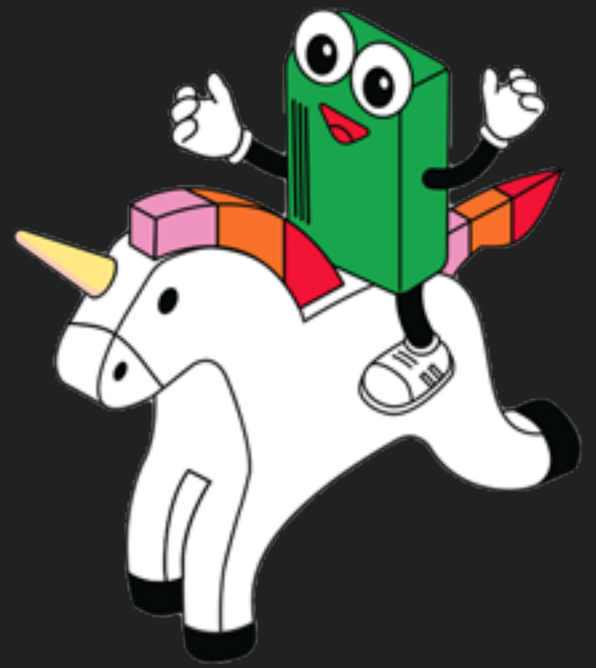
DevSecOps and Secure Incident Response

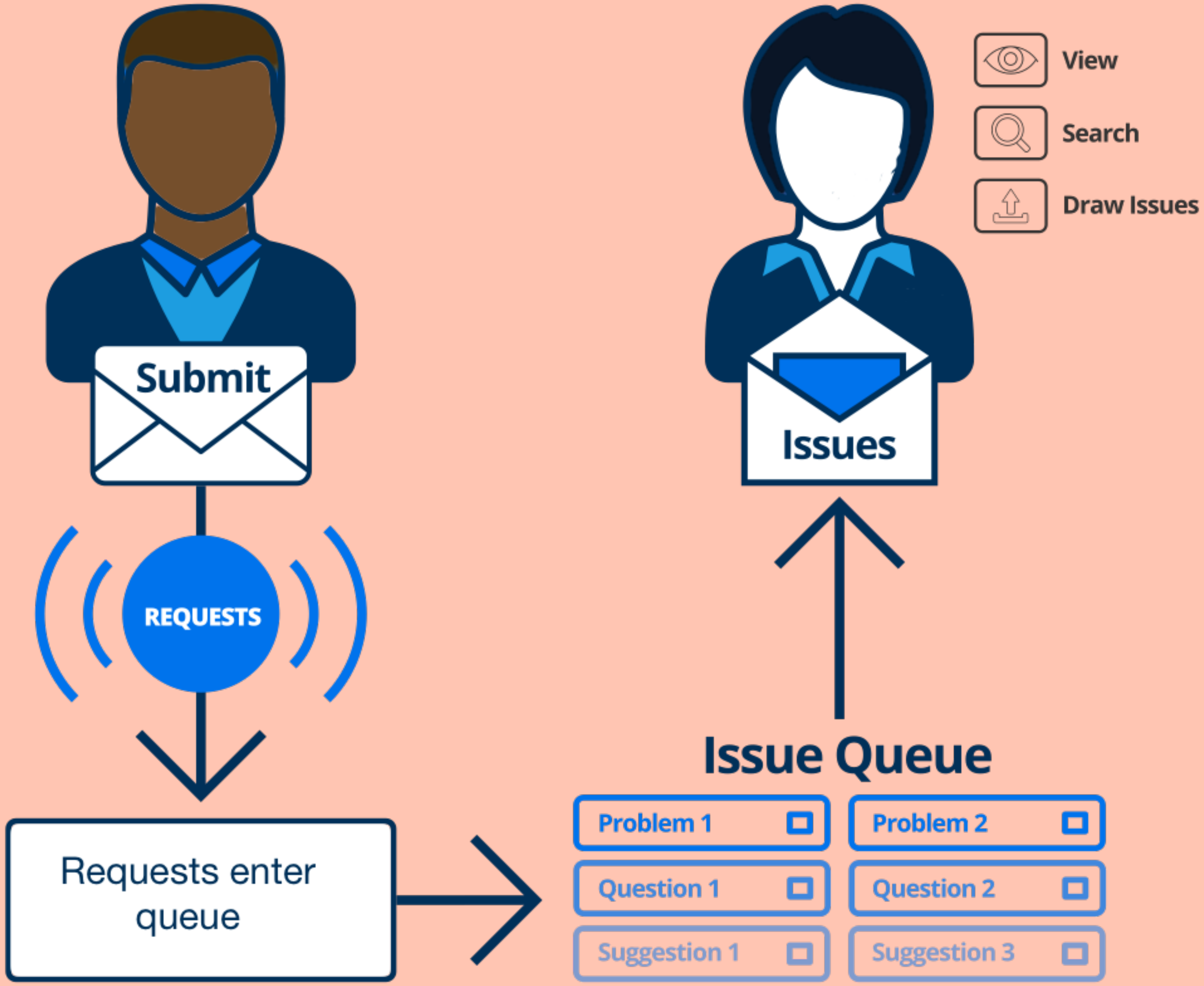
@QuintessenceAnx

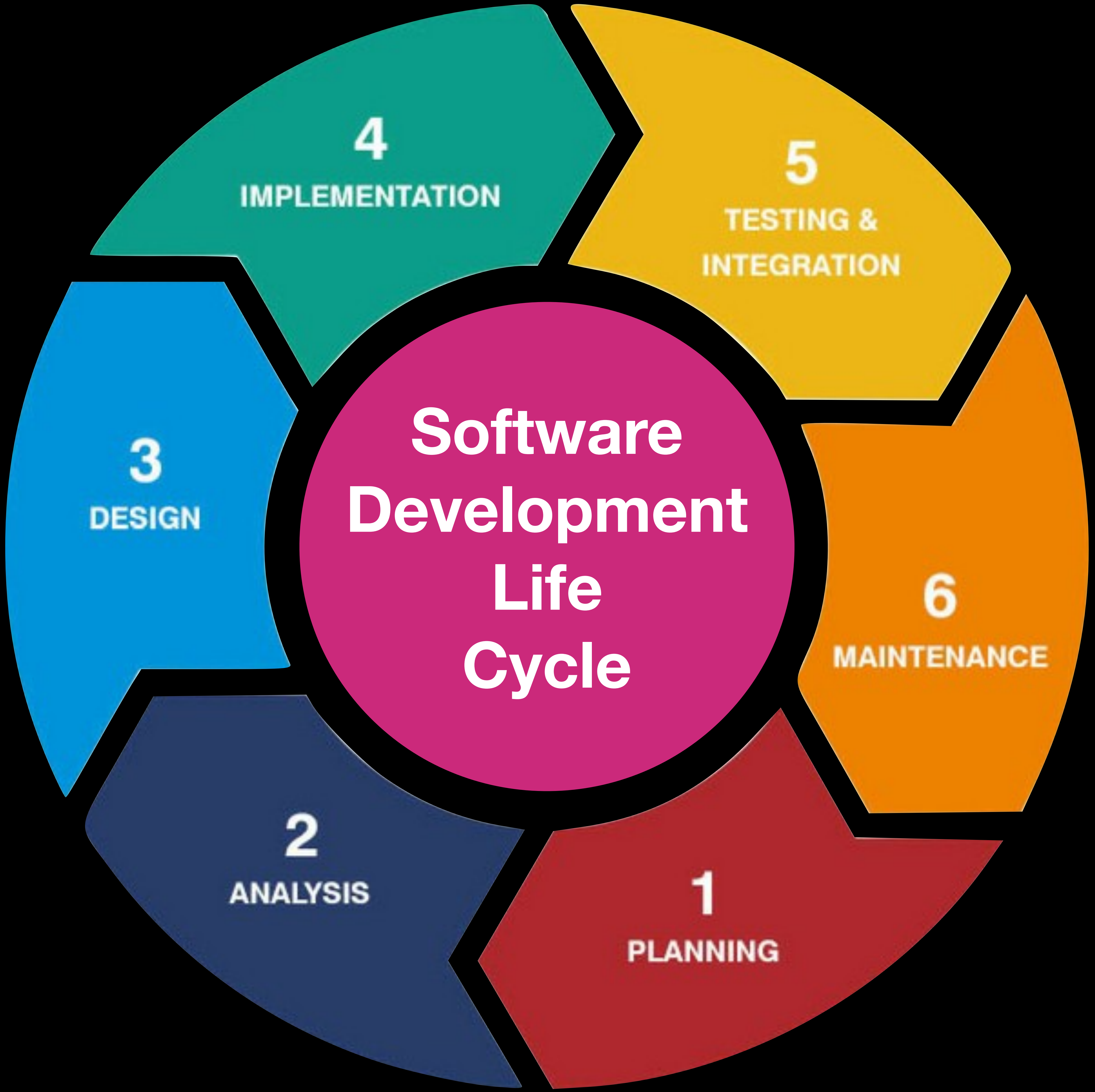
Developer Advocate @ PagerDuty



Don't panic

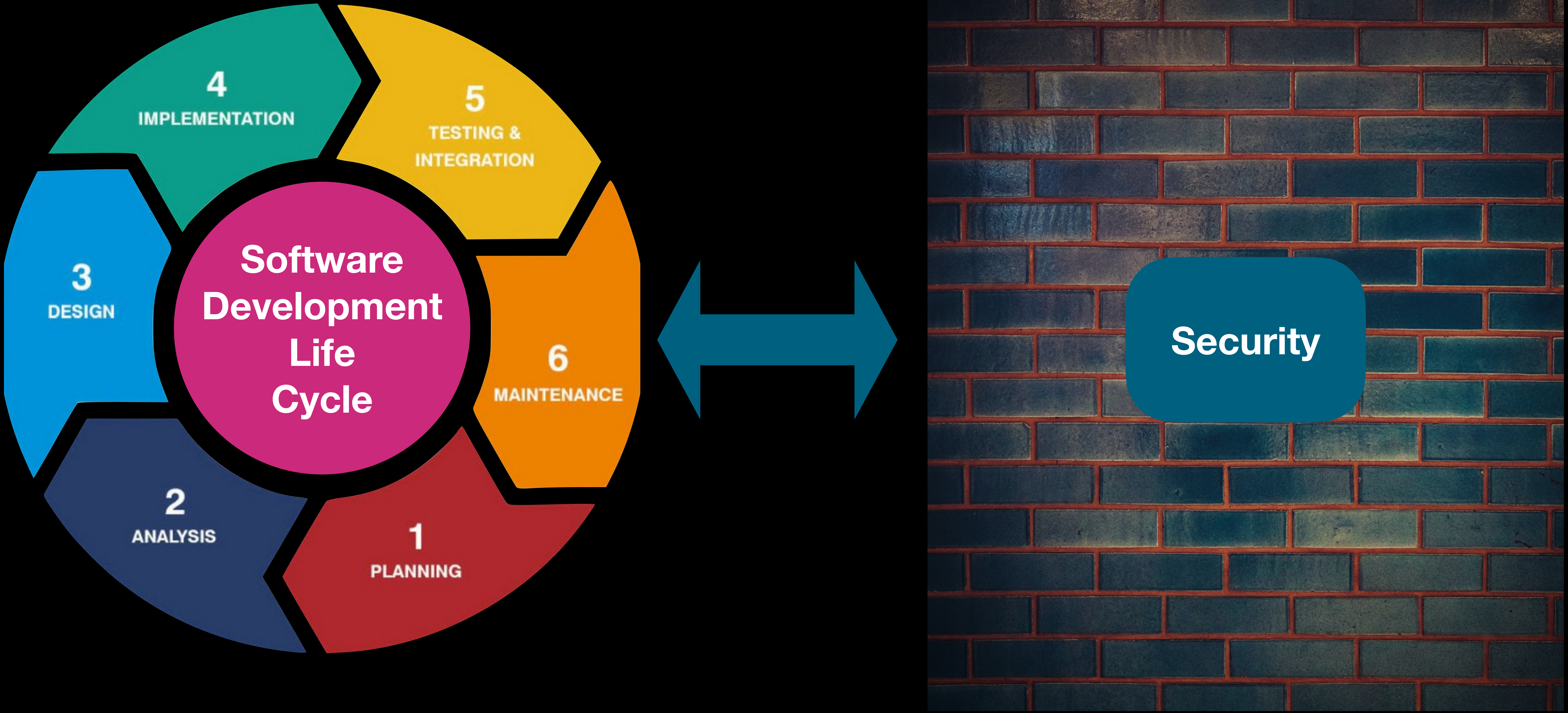
The Now

View
Search
Draw Issues
Submit
Issues
REQUESTS
Issue Queue
Requests enter queue
Problem 1
Problem 2
Question 1
Question 2
Suggestion 1
Suggestion 3


Software Development Life Cycle
1
PLANNING
2
ANALYSIS
3
DESIGN
4
IMPLEMENTATION
5
TESTING &
INTEGRATION
6
MAINTENANCE

Vault over "The Wall"

for
Security Review


Software Development Life Cycle
1
PLANNING
2
ANALYSIS
3
DESIGN
4
IMPLEMENTATION
5
TESTING &
INTEGRATION
6
MAINTENANCE
Security

🤔

DevSecOps

What is DevSecOps?

DevSecOps stands for development, security, and operations. DevSecOps seeks to integrate security across the SDLC and streamline the workflows between dev, sec, and ops.

What DevSecOps is **not**

DevSecOps is ***not*** replacing security with dev and/or ops, ***or*** expecting dev and/or ops to become security specialists, ***or*** expecting security to become devs and/or ops.

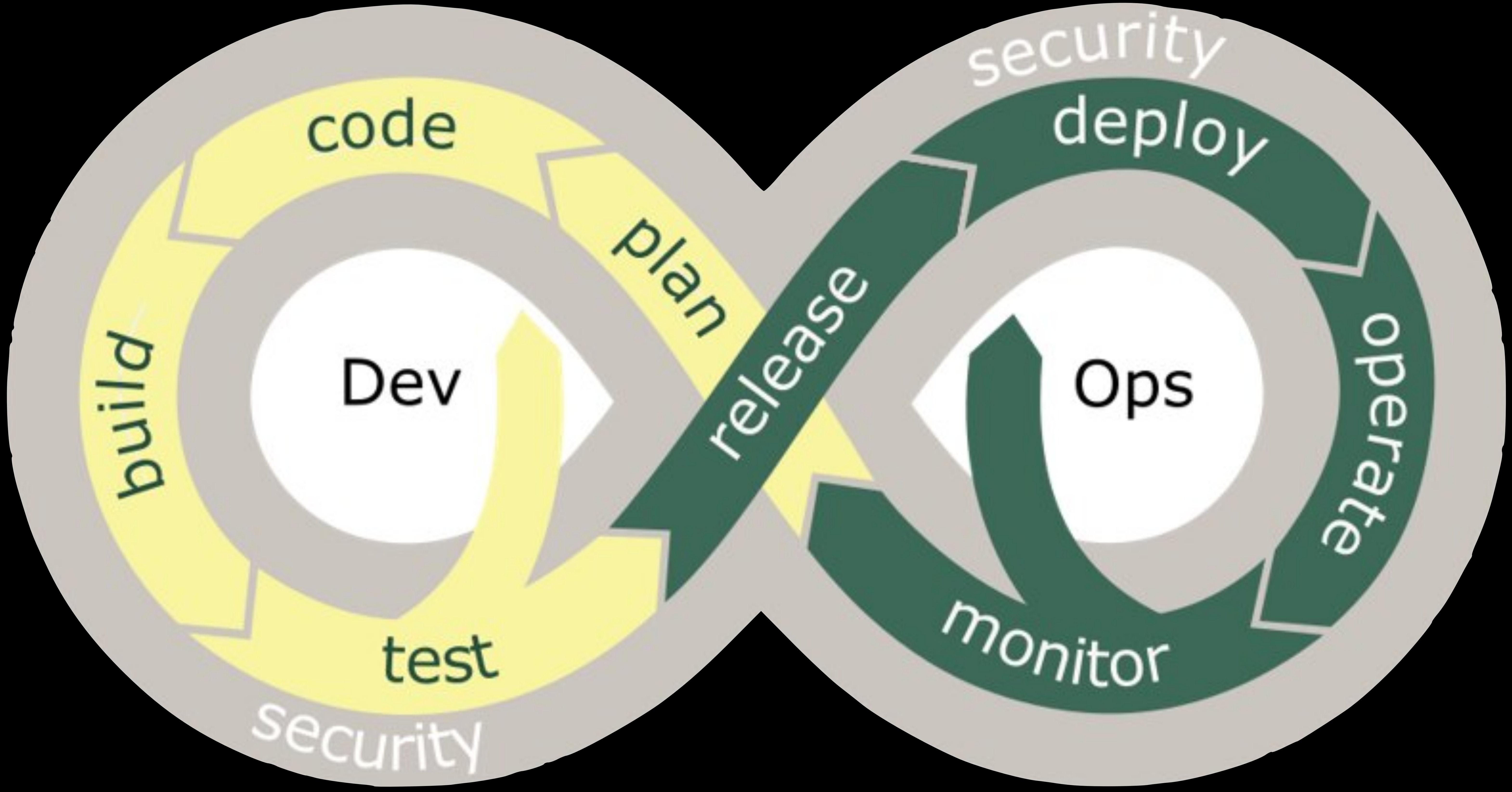
Phew.

How?

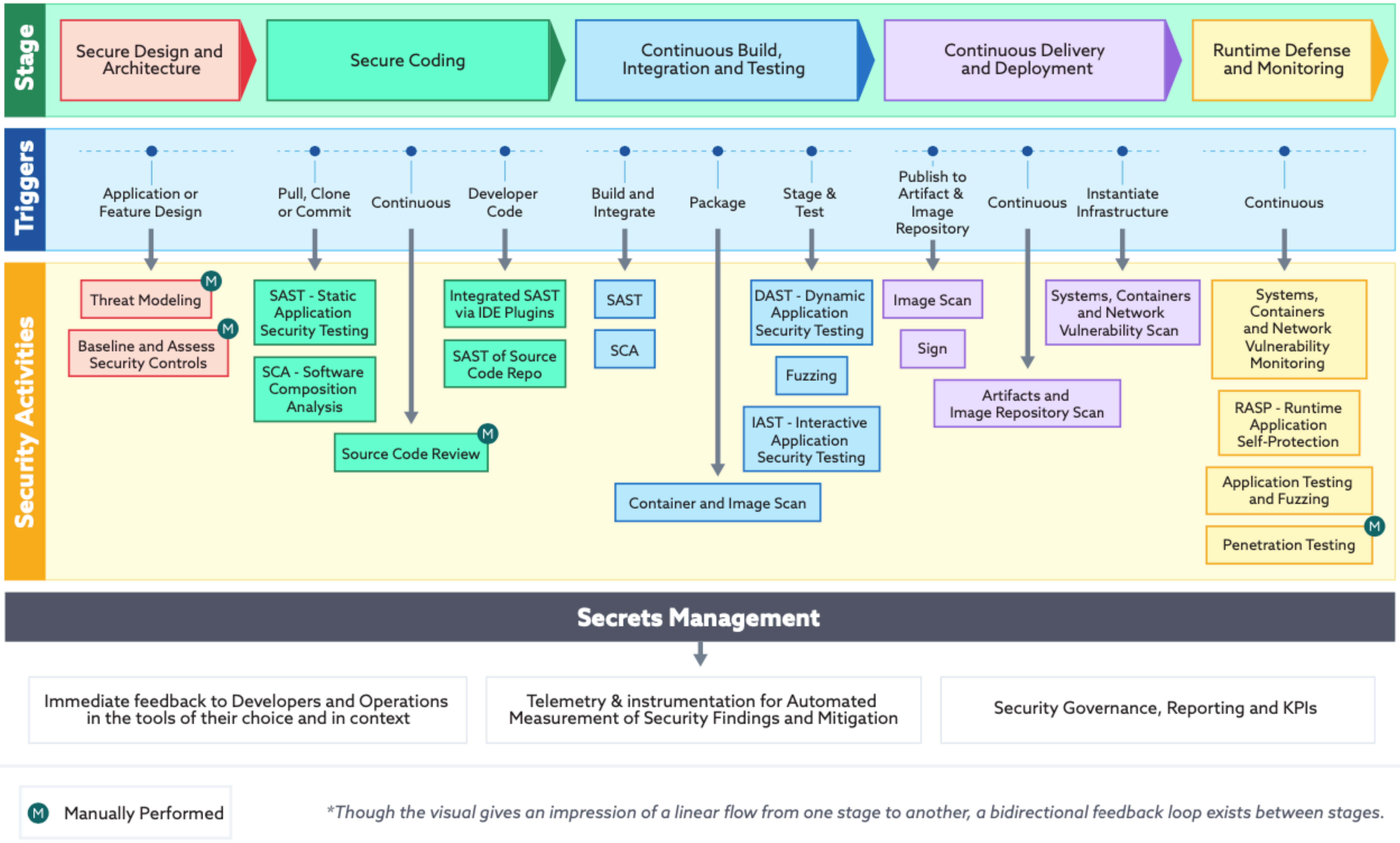
The Secure SDLC + Shifting Left

PagerDuty

@QuintessenceAnx


security
code
plan
build
Dev
test
security
release
deploy
Ops
operate
monitor

Secure Development Lifecycle - Policies, Standards, Controls and Best Practices



Stage

Secure Design and Architecture

Secure Coding

Continuous Build, Integration and Testing

Continuous Delivery and Deployment

Runtime Defense and Monitoring

Triggers

Application or Feature Design

Pull, Clone or Commit

Continuous

Developer Code

Build and Integrate

Package

Stage & Test

Publish to Artifact & Image Repository

Continuous

Instantiate Infrastructure

Continuous

Security Activities

Threat Modeling (M)

Baseline and Assess Security Controls (M)

SAST - Static Application Security Testing

SCA - Software Composition Analysis

Source Code Review (M)

Integrated SAST via IDE Plugins

SAST of Source Code Repo

SAST

SCA

Container and Image Scan

DAST - Dynamic Application Security Scan

Fuzzing

IAST - Interactive Application Security Testing

Image Scan

Sign

Artifacts and Image Repository Scan

Systems, Containers and Network Vulnerability Scan

Systems, Containers and Network Vulnerability Monitoring

RASP - Runtime Application Self-Protection

Application Testing and Fuzzing

Penetration Testing (M)

Secrets Management

Immediate feedback to Developers and Operations in the tools of their choice and in context

Telemetry & instrumentation for Automated Measurement of Security Findings and Mitigation

Security Governance, Reporting and KPIs

(M) Manually Performed

**Though the visual gives an impression of a linear flow from one stage to another, a bidirectional feedback loop exists between stages.*

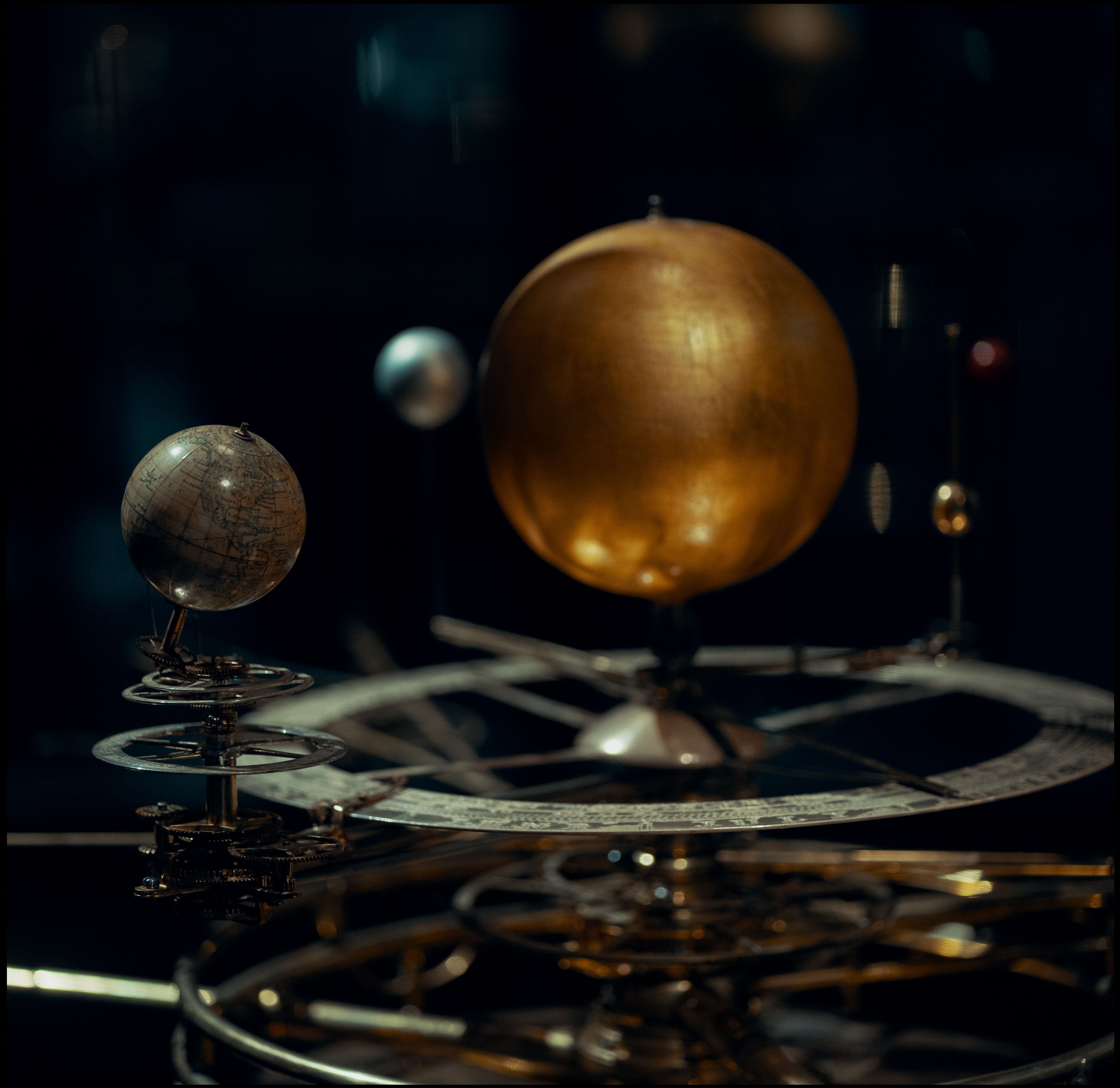
Figure 1: The CSA DevSecOps Delivery Pipeline

PagerDuty

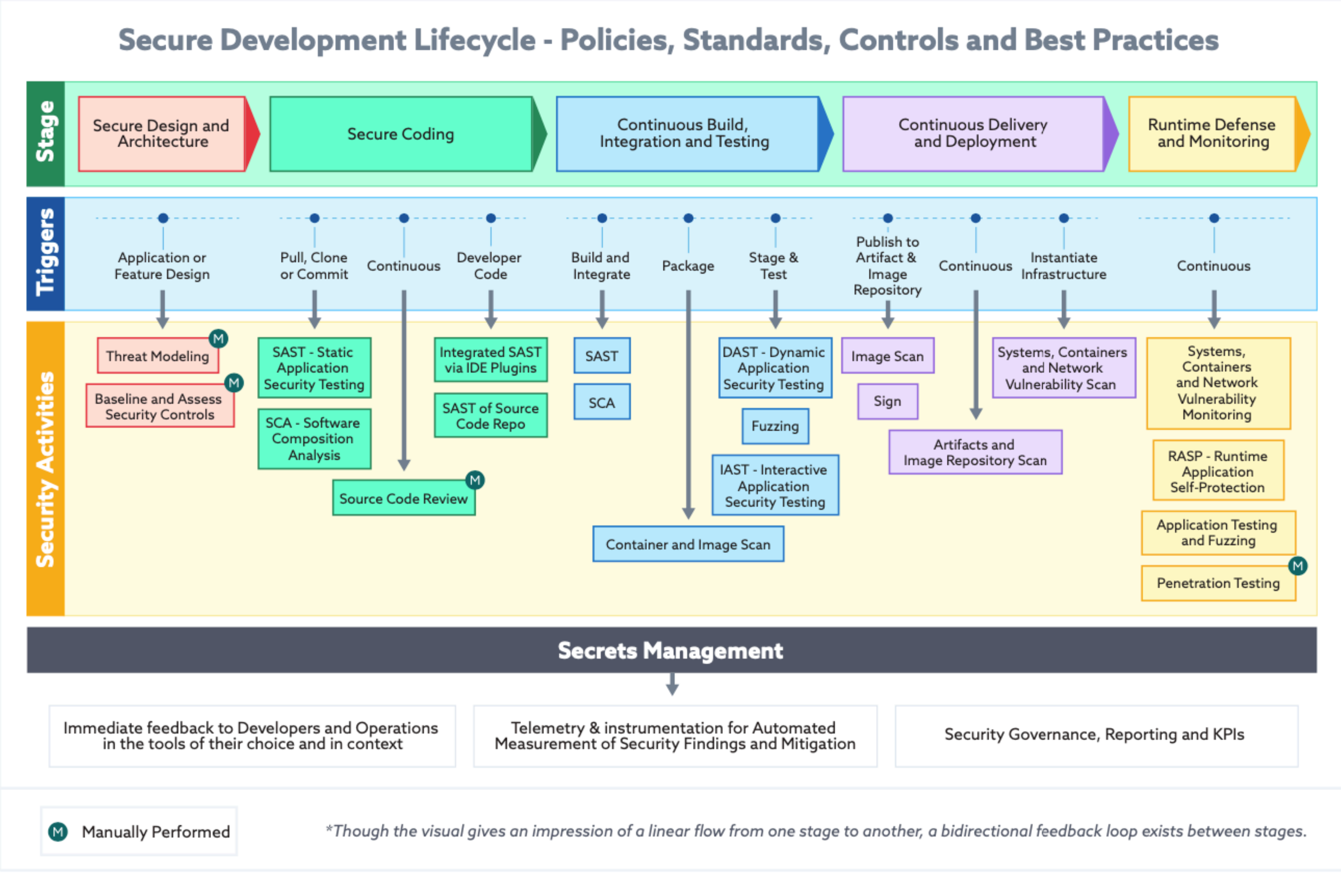
@QuintessenceAnx

SecOps Activities

- Secure architecture / design
- Threat modeling
- Testing, e.g. SAST and DAST
- Scanning images and dependencies
- Fuzzing
- And more!



Shift Left



Secure Development Lifecycle - Policies, Standards, Controls and Best Practices

Stage

Secure Design and Architecture

Secure Coding

Continuous Build, Integration and Testing

Continuous Delivery and Deployment

Runtime Defense and Monitoring

Triggers

Application or Feature Design

Pull, Clone or Commit

Continuous

Developer Code

Build and Integrate

Package

Stage & Test

Publish to Artifact & Image Repository

Continuous

Instantiate Infrastructure

Continuous

Security Activities

Threat Modeling M

Baseline and Assess Security Controls M

SAST - Static Application Security Testing

SCA - Software Composition Analysis

Source Code Review M

Integrated SAST via IDE Plugins

SAST of Source Code Repo

SAST

SCA

Container and Image Scan

DAST - Dynamic Application Security Testing

Fuzzing

IAST - Interactive Application Security Testing

Image Scan

Sign

Artifacts and Image Repository Scan

Systems, Containers and Network Vulnerability Scan

Systems, Containers and Network Vulnerability Monitoring

RASP - Runtime Application Self-Protection

Application Testing and Fuzzing

Penetration Testing M

Secrets Management

Immediate feedback to Developers and Operations in the tools of their choice and in context

Telemetry & instrumentation for Automated Measurement of Security Findings and Mitigation

Security Governance, Reporting and KPIs

M Manually Performed

*Though the visual gives an impression of a linear flow from one stage to another, a bidirectional feedback loop exists between stages.

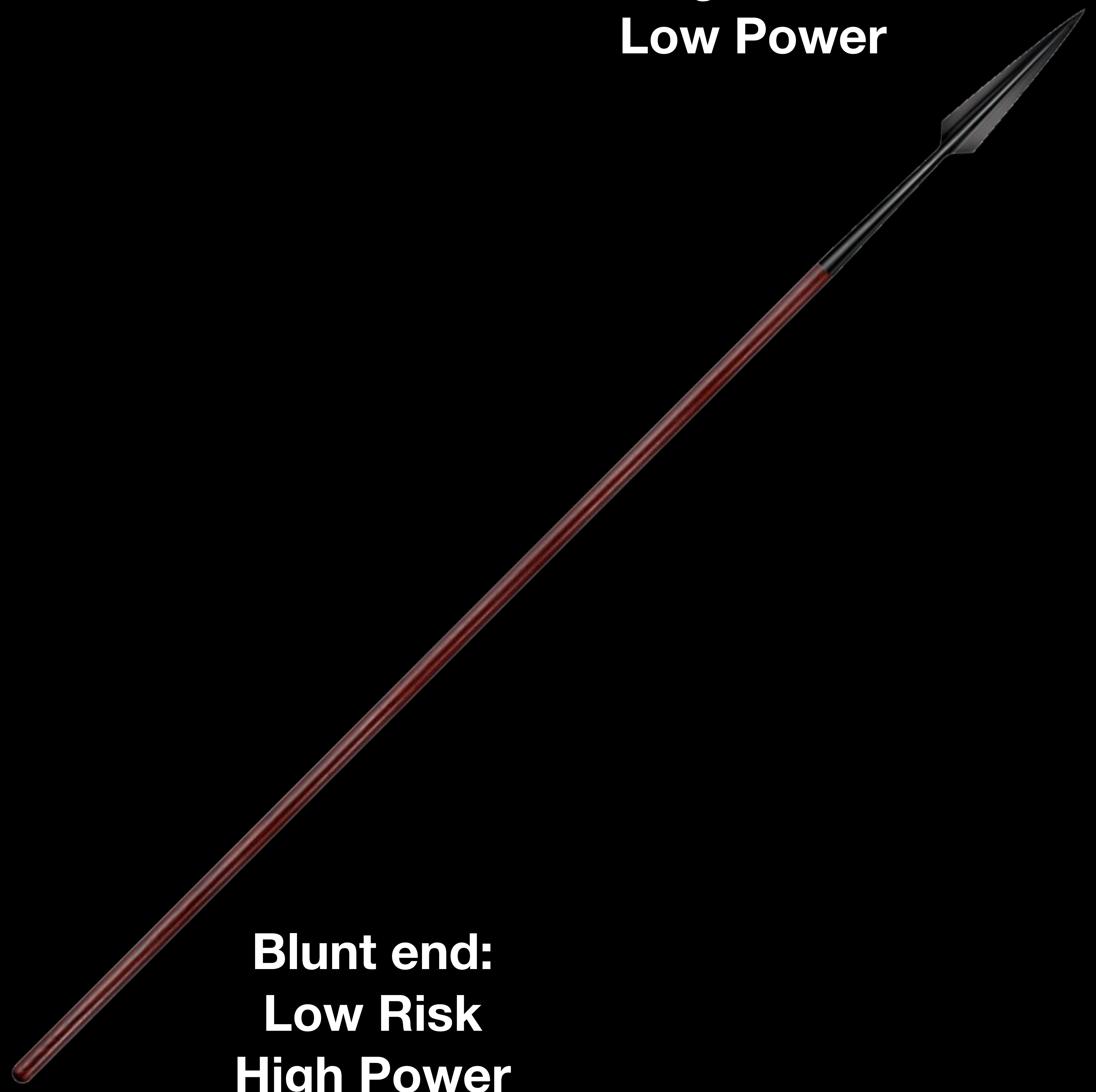
Figure 1: The CSA DevSecOps Delivery Pipeline

Precious Mini Minions
NAILED IT

How?

Cultural Support

Humans.



Sharp end:
High Risk
Low Power

Blunt end:
Low Risk
High Power

Exec Buy-in

Never trick staff, ***ever***.

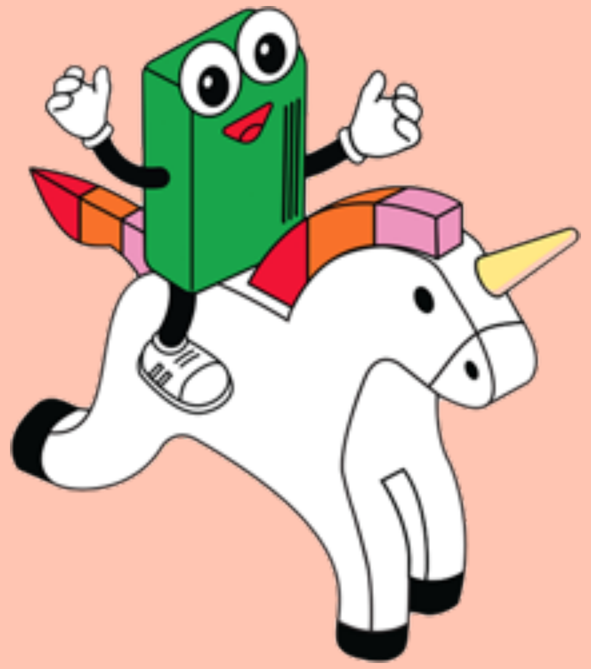
Training

Full Service Ownership

Capture the Flag

Threat Modeling

Secure Incident Response



1. Stop the attack in progress.
2. Cut off the attack vector.
3. Assemble the response team.
4. Isolate affected instances.
5. Identify timeline of attack.
6. Identify compromised data.
7. Assess risk to other systems.
8. Assess risk of re-attack.
9. Apply additional mitigations, make changes to monitoring, etc.
10. Forensic analysis of compromised systems.
11. Internal communication.
12. Involve law enforcement.
13. Reach out to external parties that may have been used as vector for attack.
14. External communication.

Stop the attack in progress

Cut off the attack vector

Assemble the response team

Isolate the affected instances

Identify timeline of the attack

Identify compromised data

Assess risk to other systems

Assess risk of re-attack

Apply additional mitigations, additions to monitoring, etc.

Forensic analysis of compromised systems

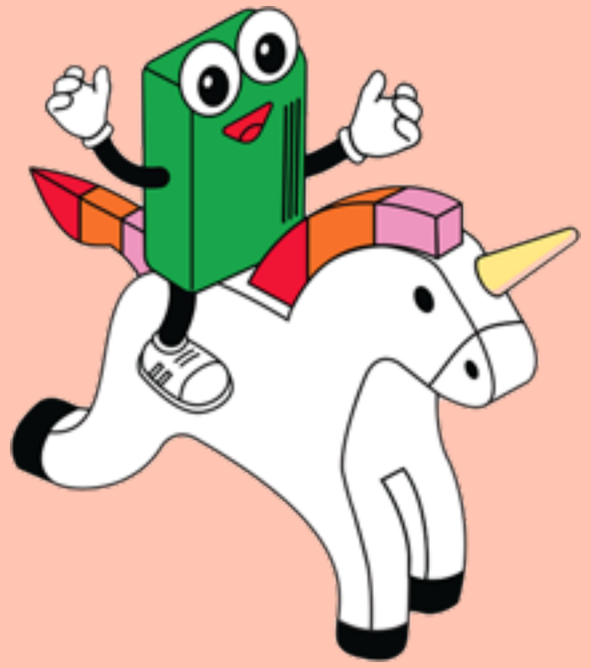
Internal communication

Involve law enforcement

Reach out to external parties that may have been used as attack vectors

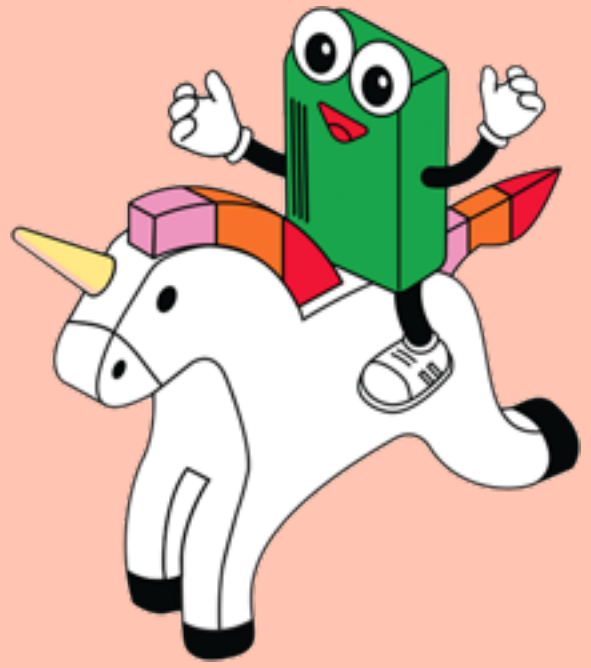
External communication

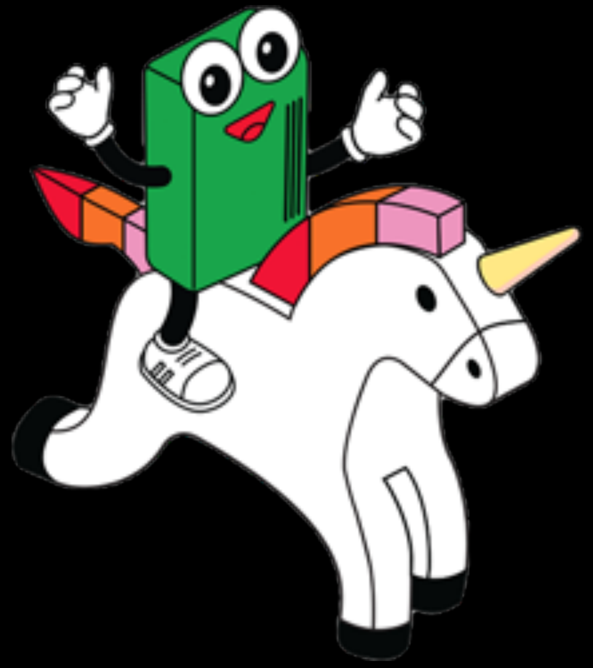
1. Stop the attack in progress.
2. Cut off the attack vector.
3. Assemble the response team.
4. Isolate affected instances.
5. Identify timeline of attack.
6. Identify compromised data.
7. Assess risk to other systems.
8. Assess risk of re-attack.
9. Apply additional mitigations, make changes to monitoring, etc.
10. Forensic analysis of compromised systems.
11. Internal communication.
12. Involve law enforcement.
13. Reach out to external parties that may have been used as vector for attack.
14. External communication.



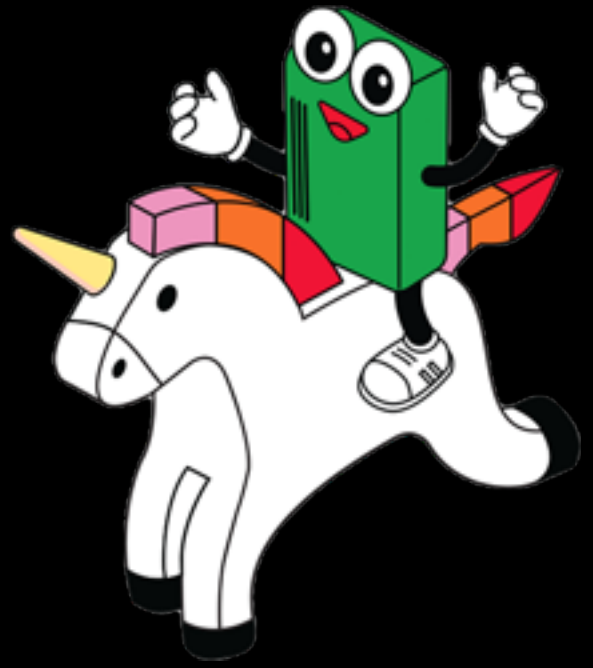
Resources & References

noti.st/quintessence





PagerDuty Summit
22-25 June



Register: http://bit.ly/PDsummitCAD

Questions?

Quintessence Anx
Developer Advocate

PagerDuty

noti.st/quintessence