snyk

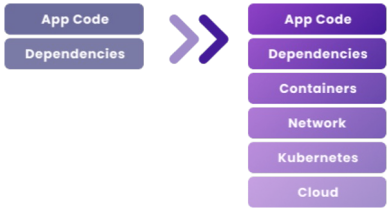
Self-Governing DevSecOps: Navigating Towards Continuous Security

Lawrence Crowther
Head of Solution Engineering, APJ

The nature and speed of development continues to change

Distributed dev teams introducing code with growing frequency

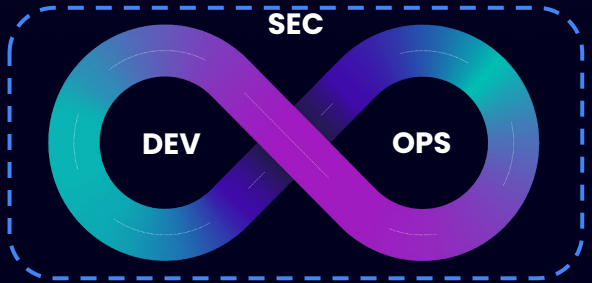
Increasingly complex software supply chains



Number & complexity of issues in backlogs continues to grow

Gas to Electric

=

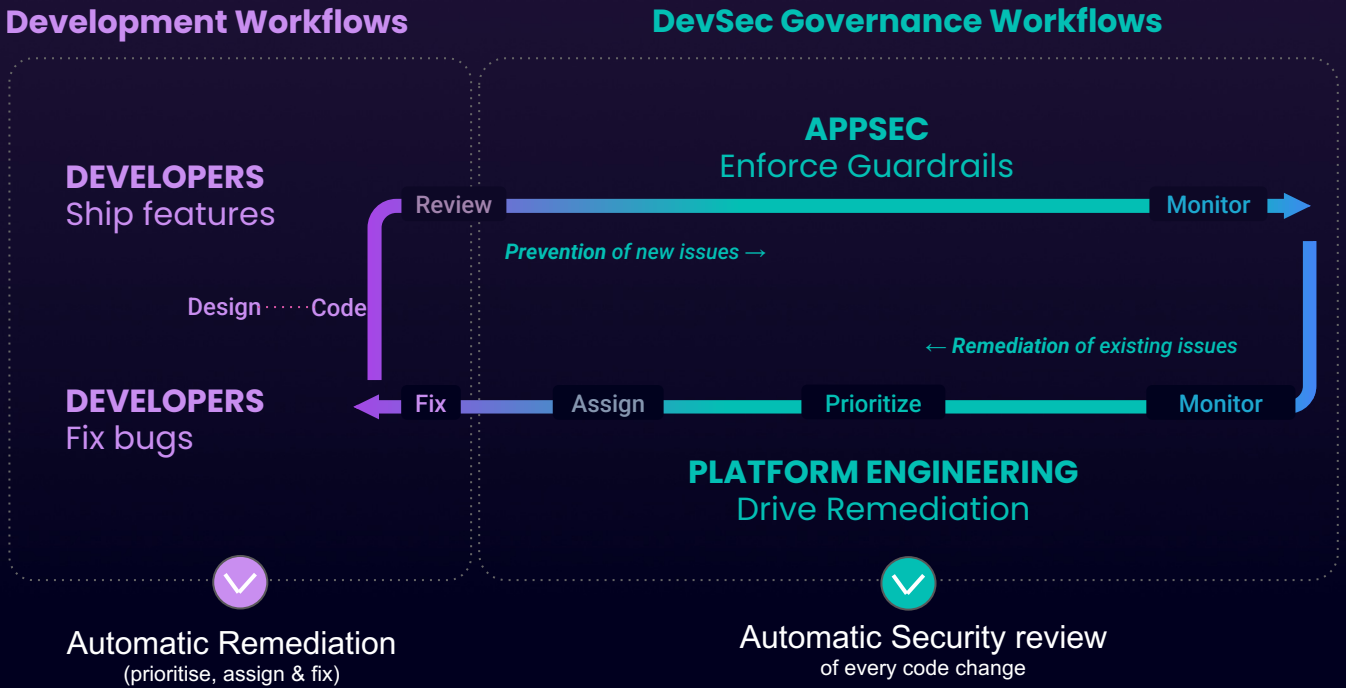


snyk

Electric to Autonomous

= ?

Self-Governing DevSecOps



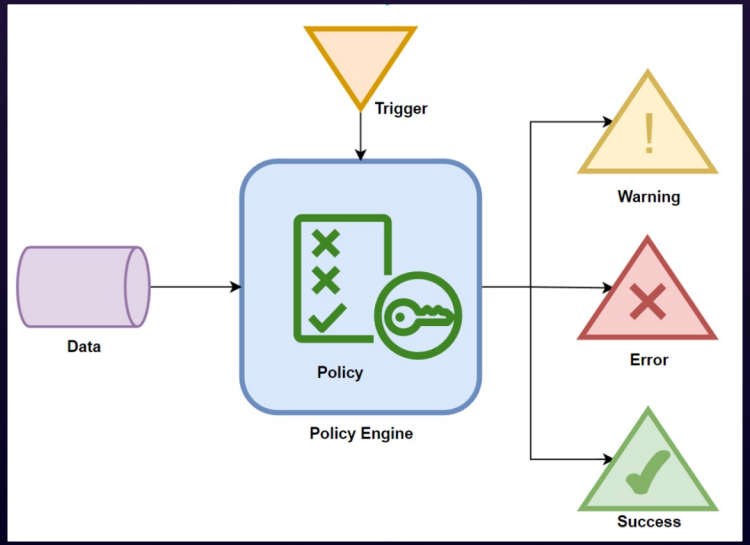
Combines policies, with workflow, automation and AI enabled learning

snyk

3 Pillars of Self-Governing DevSecOps

1. Policy As Code
2. Use of GenAI
3. Risk Focused Security

1. Policy as Code



Policy as Code in DevSecOps is about treating security and compliance policies with the same level of automation, integration, and version control as application code

snyk

Introducing Conftest

Conftest is a open source utility to help you write tests against structured configuration data. For instance, you could write tests for your Kubernetes configurations
https://www.conftest.dev/

```
package main

deny[msg] {
  input.kind == "Deployment"
  not input.spec.template.spec.securityContext.runAsNonRoot

  msg := "Containers must not run as root"
}

deny[msg] {
  input.kind == "Deployment"
  not input.spec.selector.matchLabels.app

  msg := "Containers must provide app label for pod selectors"
}
```

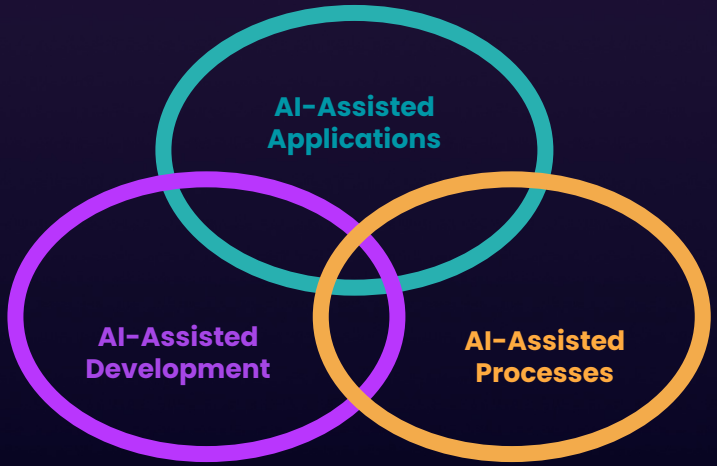
snyk

More Conftest Examples

- SAST scan failing on at least one of SQL Injection or Hardcoded Password
- Using a count strategy against number of like vulnerabilities found

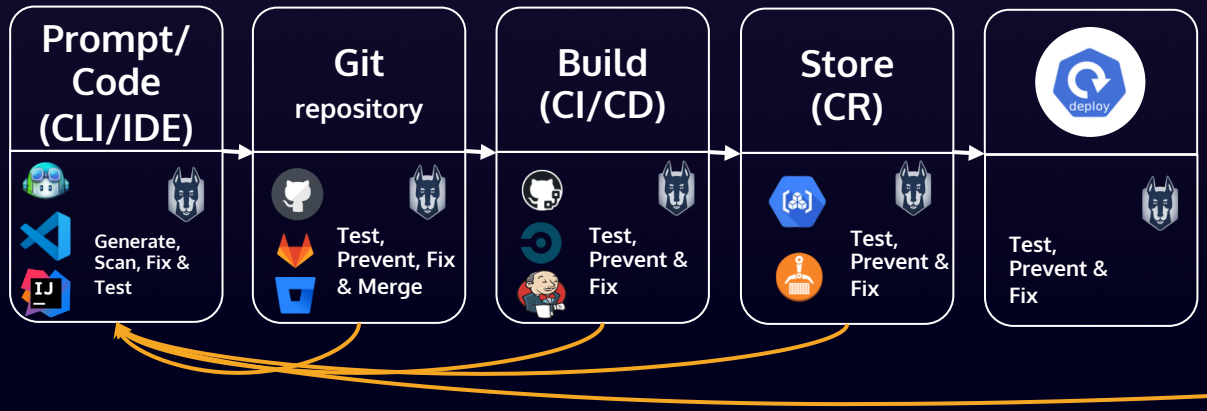
```
package main

# Set these to the number you require. The policy will fail if it finds a single vulverability defined as either of the following
rule_names = {
  "HardcodedPassword": 0,
  "Sqli": 0
}

deny[msg] {
  rule = rule_value
  rule_value = rule_map[_]
  num = count([vuln | vuln = input.runs[_].tool.driver.rules[_]; vuln.name == rule_value])
  num > rule_names[rule_value]
  msg = sprintf("%s: %v is greater than the threshold of %v", [rule_value, num, rule_names[rule_value]])
}

rule_map = ["HardcodedPassword", "Sqli"]
```

2. Use of GenAI



AI-Assisted Applications: OWASP Top 10 for LLMs (i.e Prompt Injection, Input/output handling)
AI-Assisted Development: Insecure code generated from CoPilot, need security companion
A**I-Assisted Processes**: Reduce false positives, automatic fixes

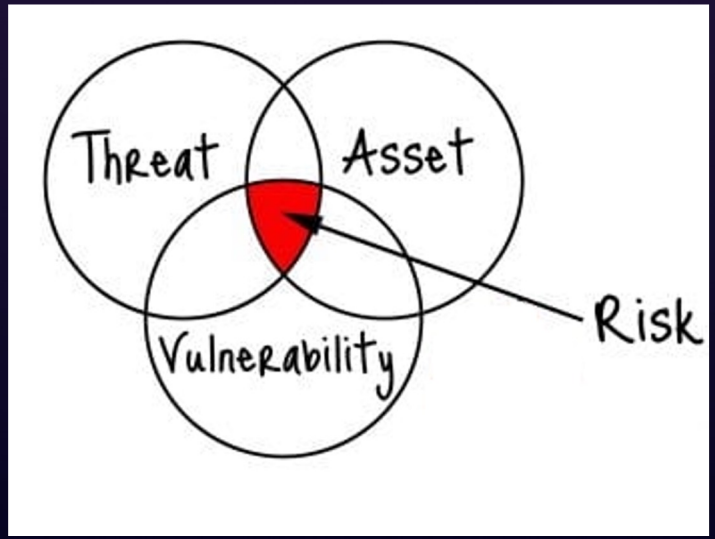
GenAI in the pipeline



Operational/Runtime Monitoring

snyk

3. Risk Focused Security



Focus on Risk vs Vulnerability management. I.e given an asset what is the real threat of exposure in production?

snyk

From shifting left to scaling up: AppSec in an era of increased complexity

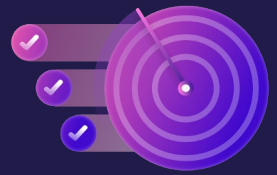
Developers and Security lack a shared lens on apps

Are all the apps covered? Are there any gaps?

Missing business and technical context

Which apps are biz-critical? Which version is alive in prod?

Where can we make meaningful impacts on risk?

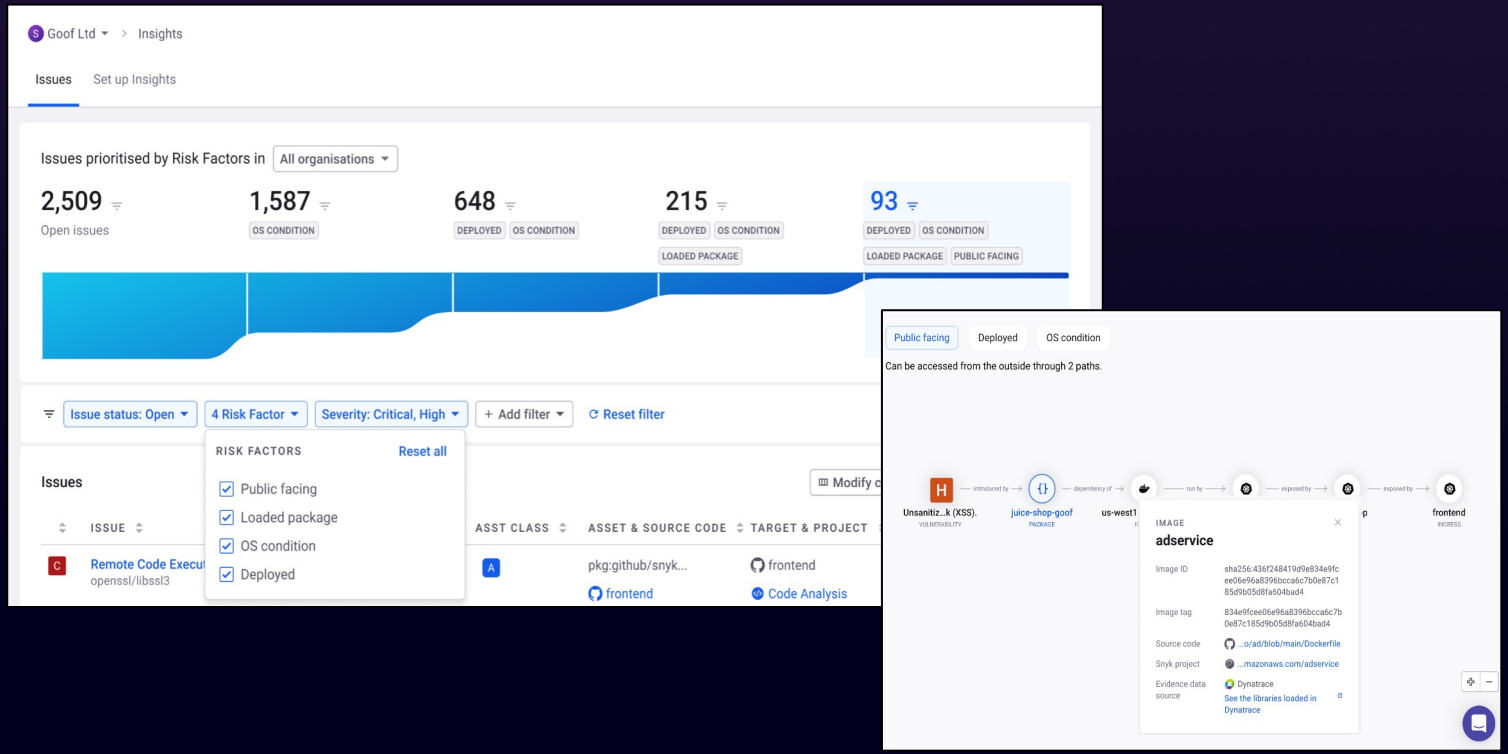


What's risk and are we truly reducing it?

snyk

Prioritise top risks

Focus your remediation efforts on what matters most to the business



snyk

DevSec Governance Platform

Policy & Automation

Reporting & Analytics

SCA | SAST | IAC | Container

Code / IDE

IDP

Cloud

Runtime

Source Control

CI / CD

Collaboration

snyk

Business Benefits of Self-Governing DevSecOps

Policy as Code

Consistency
Efficiency
Agility

GenAI (AI-Driven Automation)

Enhanced Security
Resource Optimisation
Scalability

Risk-Focused Security

Proactive Protection
Prioritized Defense
Simpler Triage

snyk

Trust Begins with Snyk

snyk