Le métier de

DevSecOps

$ Whoami

RO!
RO!

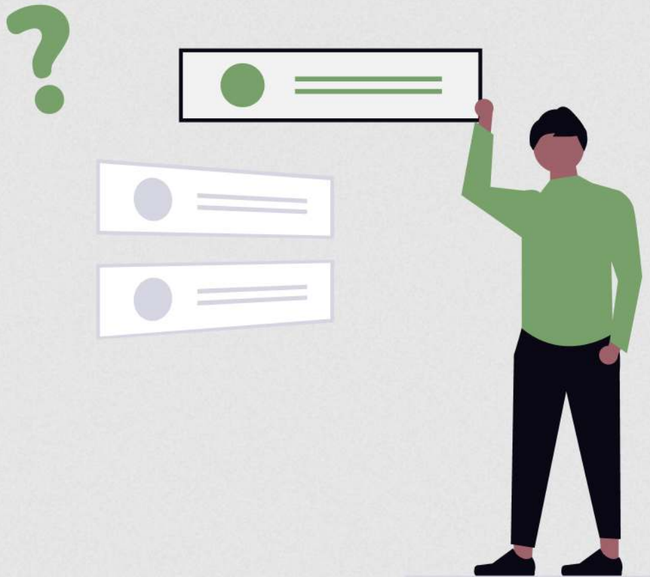
Rodolphe (aka RORO!)

- **DevSecOps;**
- **Audit de code;**
- **Pentest.**

Important

N’hésitez pas à poser des question!

Aucune question n’est mauvaise sauf les mauvaises



Introduction

L'objectif est de **créer un flux de travail fluide et continu pour livrer de la valeur plus rapidement et plus souvent**, tout en garantissant une stabilité et une qualité maximales de la production.

- **Culture de collaboration;**
- **Automatisation (Infrastructure as Code);**
- **Boucle de feedback (Observabilité);**

Introduction

Introduction

- **Intégration Continue (CI) :** Automatise la fusion du code, le build et l'exécution de tests unitaires à chaque modification. Cela permet de détecter les bugs au plus tôt et de garantir la stabilité de la branche principale.
- **Livraison/Déploiement Continu (CD) :** Automatise le passage du code validé vers les environnements de test, de pré-production ou de production. Cela réduit les interventions manuelles, limite les erreurs humaines et raccourcit le "Time-to-Market".

Exemple: Jenkins, Gitlab CI, Github Actions, XlRelease, etc...

Introduction

Shift Left?

Introduction

Shift Left?

Pourquoi?

- Détection précoce
- Correction rapide
- Responsabilité partagée

DevSec quoi?!

L'objectif est de mettre en œuvre le concept de "**Shift Left**" : déplacer les contrôles de sécurité le plus tôt possible dans le processus de création du logiciel pour identifier et corriger les vulnérabilités à moindre coût et sans ralentir la mise en production.

- Sécurité automatisée
- Infrastructure-as-Code (IaC) sécurisée
- Conformité continue

DevSec quoi?!

DevOps vs DeSecOps

	DevOps	DevSecOps
Philosophie	Unifier le développement (Dev) et les opérations (Ops) pour accélérer les livraisons.	Intégrer la sécurité (Sec) dès le début du cycle de développement (Shift Left).
Objectif Principal	Vitesse de déploiement et agilité.	Vitesse de déploiement **sécurisée**.
Responsabilité	Partagée entre développeurs et ingénieurs système.	Partagée par tous (développeurs, Ops et experts sécurité).
Rôle de la Sécurité	Souvent une étape de vérification finale (périphérique au pipeline).	Particulière et centrale à chaque étape du pipeline CI/CD.
Culture	Collaboration et automatisation des processus.	"Security is everyone's responsibility" (La sécurité est l'affaire de tous).

DevSec quoi?!

Shift Left

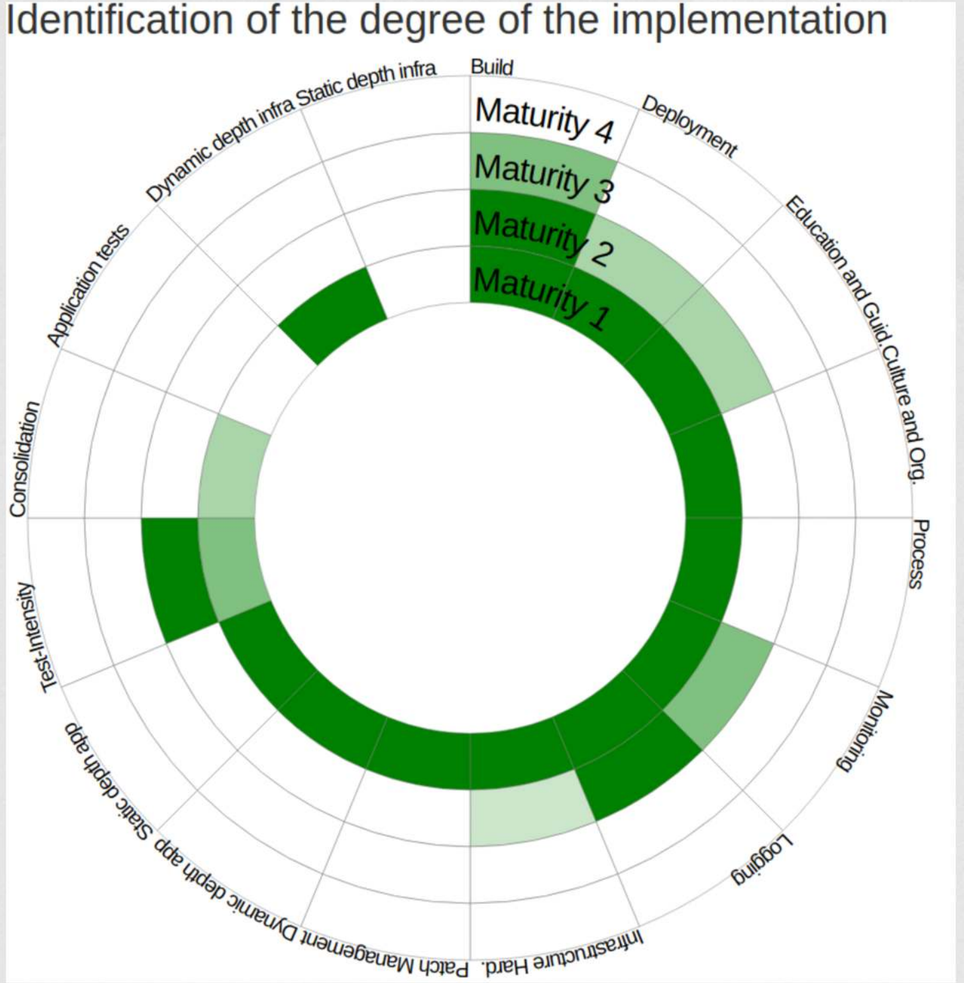
Le Shift Left, c'est l'idée de "tester tôt et tester souvent".

C'est un pilier fondamental du DevSecOps qui permet de livrer des logiciels plus fiables, plus rapidement, en évitant les mauvaises surprises de dernière minute.

DevSec quoi?!

DSOMM

DevSec quoi?!



Ok mais comment??

Ok mais comment??

Détection des secrets

Le danger des secrets

- Le Mythe du Dépôt Privé
- La Persistance de Git

Conséquences

- Leak de l'IP
- Accès aux données sensible

Ok mais comment??

Détection des secrets

Le danger des secrets

- Le Mythe du Dépôt Privé
- La Persistance de Git

Conséquences

- Leak de l'IP
- Accès aux données sensible

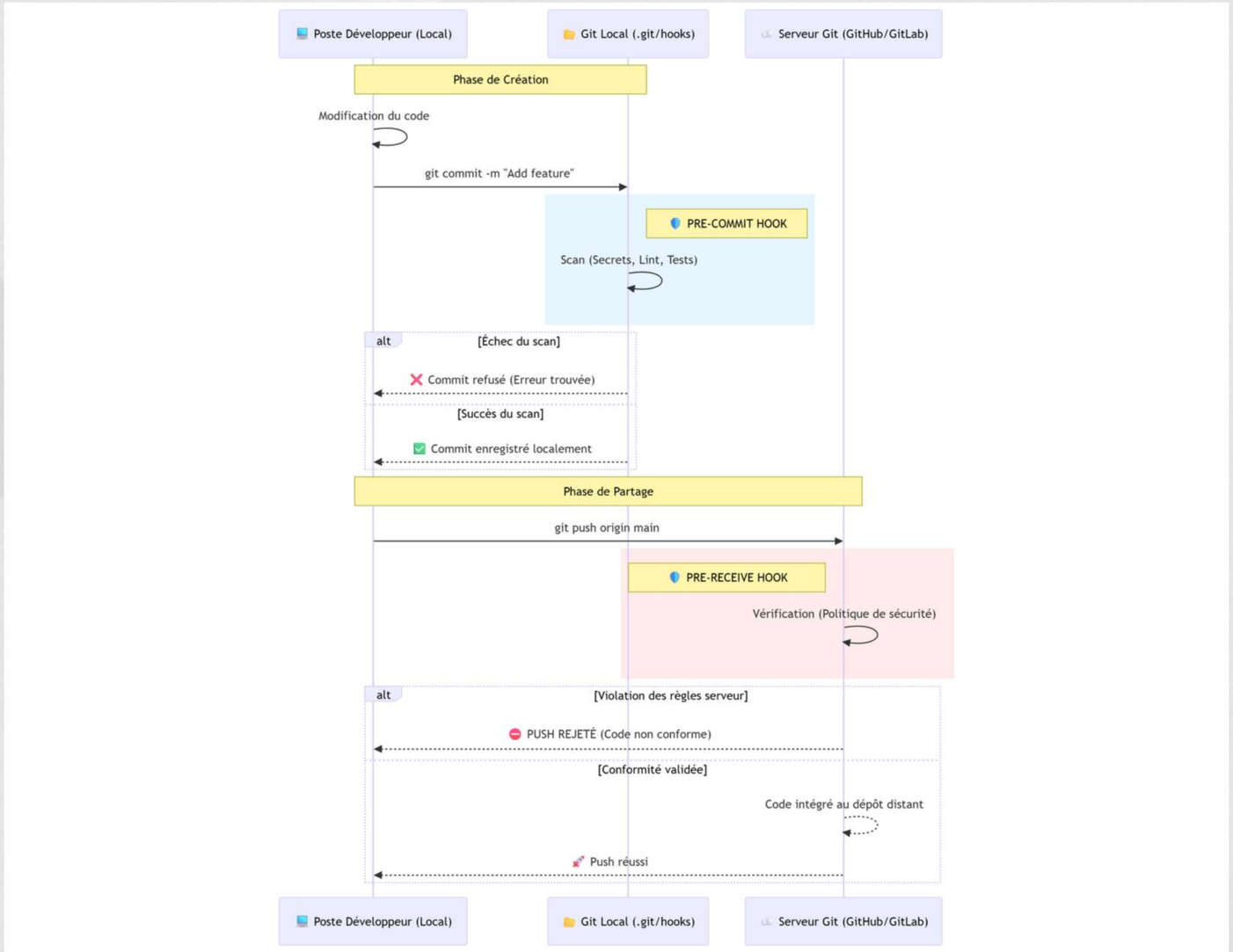
Prévention

- Pre commit hook
- Pre receive hook
- Détection de secret dans la CI/CD (un peu tard)

Tools

- Gitleaks (secret detection)
- TruffleHog (secret detection)
- Vault (secret manager)
- OpenBao (secret manager)

Ok mais comment??



Ok mais comment??

Analyse des dépendances

Le SCA (Software Composition Analysis) consiste à analyser les bibliothèques externes que l'on importe (npm, PyPI, Maven).

- Le problème de la Supply Chain : Quand vous installez un package, vous installez aussi ses propres dépendances."
- Le concept de "Vulnerability Transit" : Vous utilisez une librairie saine, qui utilise une librairie saine, qui utilise une librairie vulnérable.
- Outils :
  - npm audit (Node.js)
  - GitHub Dependabot
  - Trivy
  - Snyk

Ok mais comment??

Analyse des dépendances

```
trivy fs --scanners vuln . -q

Report Summary

┌──────────────────┬──────┬─────────────────┐
│      Target      │ Type │ Vulnerabilities │
├──────────────────┼──────┼─────────────────┤
│ requirements.txt │ pip  │        4        │
└──────────────────┴──────┴─────────────────┘

requirements.txt (pip)
Total: 4 (LOW: 0, MEDIUM: 3, HIGH: 0, CRITICAL: 1)

┌─────────────┬─────────────────┬──────────┬───────────────────┬───────────────┐
│   Library   │ Vulnerability   │ Severity │ Installed Version │ Fixed Version │
├─────────────┼─────────────────┼──────────┼───────────────────┼───────────────┤
│ python-jose │ CVE-2024-33663  │ CRITICAL │ 3.3.0             │ 3.4.0         │
│             ├─────────────────┼──────────┤                   │               │
│             │ CVE-2024-33664  │ MEDIUM   │                   │               │
│             ├─────────────────┤          │                   │               │
│             │ CVE-2016-1000   │          │                   │               │
├─────────────┼─────────────────┤          ├───────────────────┼───────────────┤
│ requests    │ CVE-2024-35195  │ MEDIUM   │ 2.32.3            │ (no fix)      │
└─────────────┴─────────────────┴──────────┴───────────────────┴───────────────┘
```

Ok mais comment??

Analyse du code source

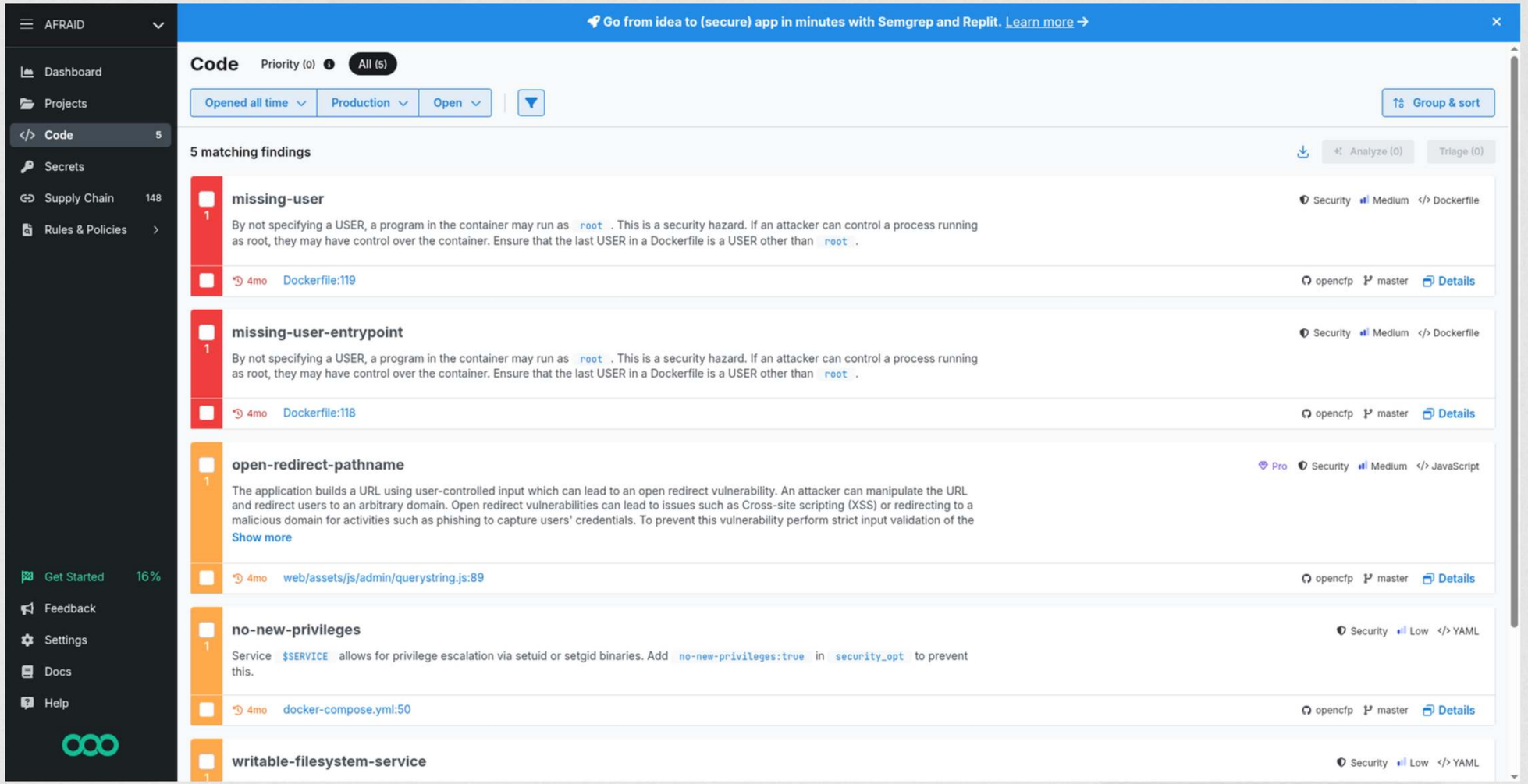
Le SAST (Static Application Security Testing) est comme un linter (comme ESLint ou Sonar), mais spécialisé dans la détection de faiblesse logiciel.

- **Précocité** : On détecte l'erreur pendant que le développeur écrit le code (Shift Left).
- **Couverture** : L'outil scanne 100% des lignes de code, même celles qui ne sont jamais exécutées en test.
- **Éducation** : L'outil explique généralement pourquoi la ligne est dangereuse (ex: "Utilisation d'une fonction de hachage faible MD5").
- **Outils** : Semgrep, Snyk, bandit, Checkmarks, Fortify.

(Attention aux faux positifs)

Ok mais comment??

Analyse du code source

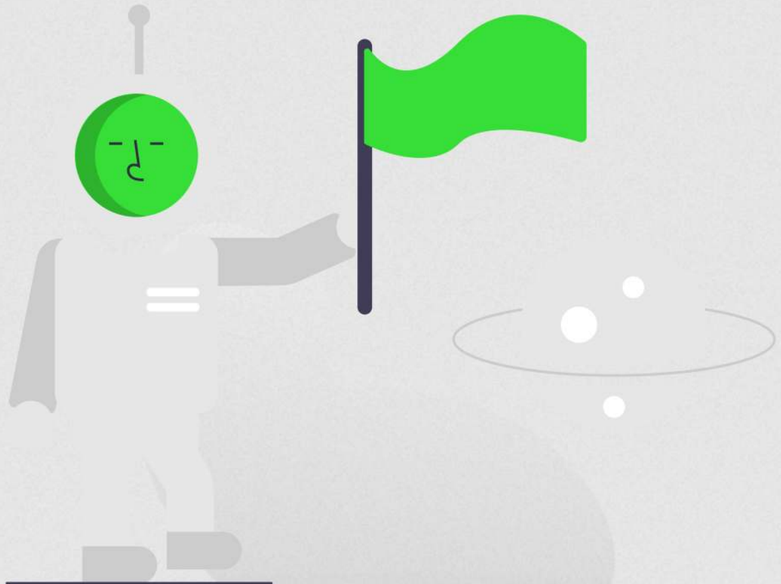


Ok mais comment??

Analyse du code source

Comment bien choisir son SAST?

- **Couverture des languages de la codebase**
  - % de couverture?
  - Compréhesion des frameworks utilisés?
- **Performance**
  - Buildless ?
  - Temps de scan ?
- **Intégration ?**
  - CI/CD (setup, plug-in ?)
  - Extension IDE ?



Ok mais comment??

Analyse du code source

Les Red flags

- Documentation moyenne
- Pas d’API
- Documentation derrière un paywall
- Peu de language supporté

Ok mais comment??

ASPM

L’ASPM est une approche et une catégorie d’outils visant à centraliser, corréler et orchestrer toutes les données de sécurité liées aux applications tout au long de leur cycle de vie.

L’objectif est de fournir une visibilité globale, contextualisée et exploitable de l’état de sécurité d’une application.

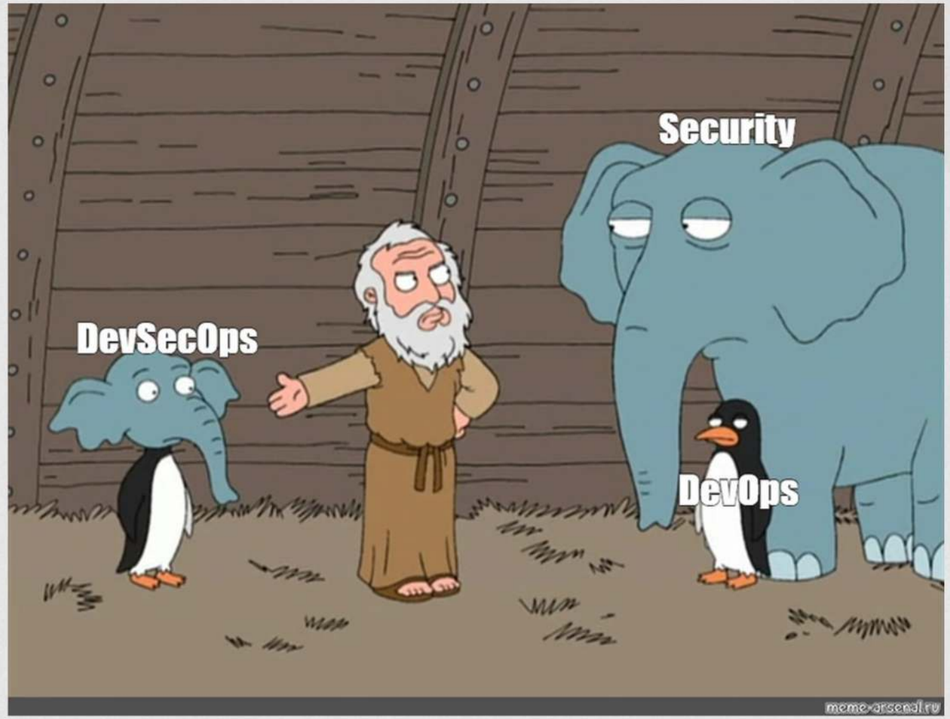
Pourquoi ?:

Il permet de centraliser l’ensemble des signaux de sécurité, de les croiser avec des informations contextuelles (exposition Internet, rôle de l’application, SBOM, etc.) et de faciliter la collaboration entre les équipes Dev, Sec et Ops.

Dans la vrai vie

Tâches variées:

- Intégration d'outils
- Collaboration entre plusieurs équipes
- Stack technique variés
- Un pied dans la sécurité
- Un pied dans le DevOps



Merci!