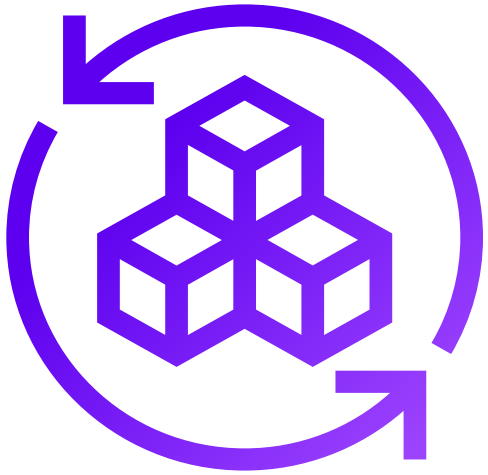
AWS + Datadog

Sécuriser l'angle mort de votre pipeline

Daniel « *phrawzty* » Maher
Senior Security Advocate & Researcher



Vous savez ce qui entre dans votre application ?





Ce que les données nous disent

	npm	PyPI	RubyGems	crates.io	Hex.pm	Packagist	*Max*
Auth (Human)	3	3	0	1	0	0	*4*
Auth (Machine)	6	4	4	4	2	2	*7*
Access Control	6	4	4	4	5	4	*7*
Build & Provenance	6	4	3	2	2	2	*8*
Integrity & Signing	5	5	6	2	3	2	*6*
Vulnerability Mgmt	7	3	4	3	4	4	*7*
Abuse Prevention	6	5	5	5	5	2	*6*
Transparency	5	4	5	3	3	2	*7*

Les lacunes qui comptent

- **Authentification**
  - un contrôle optionnel n'est pas un contrôle
- **Provenance**
  - aucune garantie que le paquet vient de sa source
- **Build**
  - aucun registre n'impose un environnement de build isolé

Les lacunes qui comptent

- **Authentification**
  - un contrôle optionnel n'est pas un contrôle
- **Provenance**
  - aucune garantie que le paquet vient de sa source
- **Build**
  - aucun registre n'impose un environnement de build isolé

Les registres ne sont pas votre équipe sécurité. Ce sont des infrastructures.

L'IA amplifie le problème

- Les assistants IA ajoutent des *dépendances automatiquement*
- Le *dernier point de friction* humain vient de disparaître
- Ce qui était déjà invisible est *maintenant plus rapide*

2 h le mat …

un mainteneur compromis

---

une version malveillante publiée

---

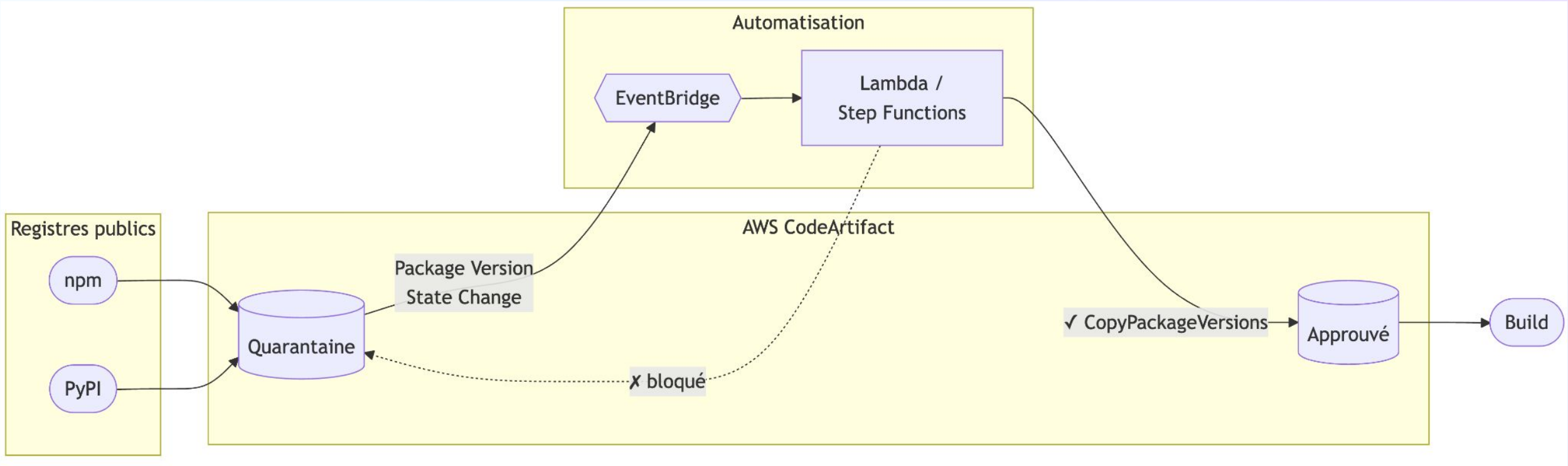
un build « `latest` » automatisé



AWS CodeArtifact : le contrôle préventif

- Service managé de dépôt d'artefacts
- Compatible avec npm, pip, Maven, NuGet, et plus encore
- Vos outils, votre workflow
- **Vous définissez ce qui entre**

Architecture de contrôle





Trois barrières

- **Isolation architecturale**
  - vos builds ne parlent jamais aux registres publics
- **Validation automatisée**
  - chaque nouvelle version est scannée avant décision
- **Promotion explicite**
  - par défaut, rien ne passe (spoiler : *version pinning*)



Trois barrières

- **Quarantaine architecturée**
  - la version malveillante n'atteint jamais vos builds
- **Épinglage de version**
  - pas de mise à jour automatique
- **Validation automatisée**
  - workflow déclenché, promotion bloquée si échec

Datadog Code Security

SCA

Software Composition Analysis

SAST

Static Code Analysis

Secret Scanning

(que ça)

IAST

Runtime Code Analysis

Datadog Code Security

SCA

Software Composition Analysis

SAST

Static Code Analysis

Secret Scanning

(que ça)

IAST

Runtime Code Analysis

6 mois plus tard …

paquet malveillante publié (oups)

---

vulnérabilité en prod (OUPS)

---

panique ?!

Six mois plus tard

- **SCA « actif »** ***(Software Composition Analysis)***
  - la bibliothèque vulnérable est détectée (service, version, CVE, …)

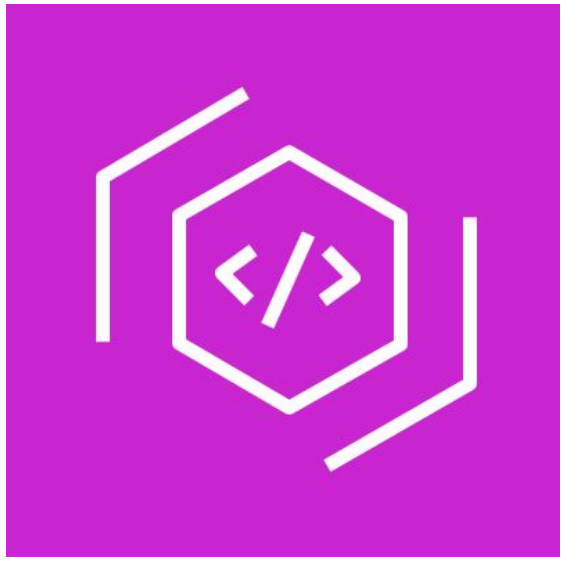
- **IAST** ***(Interactive Application Security Testing)***
  - le chemin d'appel est exercé et l'impact est confirmé

Préventif vs détectif

- **Préventif** : bloquer avant que ça entre
  - CodeArtifact, SAST, SCA (mode statique)

- **Détectif** : savoir ce qui est déjà là — *et agir*
  - SCA (mode actif), IAST

AWS et Datadog ensemble : *défense en profondeur*





Par où commencer ?

- **Cartographiez** votre graphe de dépendances — sachez ce sur quoi vous construisez
- **Identifiez** vos registres sources
- **Instaurez** une couche de contrôle préventif
- **Assurez** une visibilité totale en production

Par où commencer ?

- **Cartographiez** votre graphe de dépendances — sachez ce sur quoi vous construisez
- **Identifiez** vos registres sources
- **Instaurez** une couche de contrôle préventif
- **Assurez** une visibilité totale en production

Ce ne sont pas quatre projets sur six mois. Ce sont quatre étapes que vous pouvez commencer dès cette semaine !

Merci !

AWS CodeArtifact

`https://aws.amazon.com/codeartifact`

Datadog Code Security

`https://docs.datadoghq.com/security/code_security`

Notre blog super intéressant 😁

`https://securitylabs.datadoghq.com`