

Sécurité DevOps

Isabelle Kraemer

Ludovic Eschard

Masters SeCReTS

Mars 2020

Le devops en 2020

Julien Vehent

@jvehent

Firefox Operations Security at [@Mozilla](#) 🦸; Author of Security DevOps [securing-devops.com](#) 💻; coder, speaker, cryptstorian.



Julien Vehent @jvehent · 20 févr.

Five hours.

It took less than five hours to fix a vulnerability in a major production service from the time it was reported to us.

Five. Hours. That's it.

This shit used to take weeks!

2

2

60



LinkedIn

Devops in Paris, Ile-de-France, France
4,593 results

Job Alert Off



Session devops 9:30-11:00

COMMENT ÊTRE UN OPS DANS UNE ÉQUIPE DE DEV SANS FINIR SOUS...

Tu fais de l'OPS, et tu viens d'arriver dans une feature team ? Tu es sous l'eau entre tes tâches de réalisation, de support, et l'objectif de faire monter l'équipe en compétences ? Reste calme,...



SEBASTIÁN CACERES
ULTRA

Session devsecops 11:30-13:00

COMMENT LE DEV, L'OPS ET LA SÉCU PEUVENT MONTER DANS LE...

Intégrer la sécurité à la démarche DevOps, voilà une problématique sur laquelle de nombreuses entreprises travaillent. Au-delà d'une problématique, la Team JCDecaux y a vu un challenge !...



JOY-ALEXANDRA DENIS
JCDECAUX

DEVOPS À L'ÉCHELLE : QUELLES PRATIQUES POUR BIEN GRANDIR ?

Les pratiques DevOps fonctionnant pour les startups sont difficilement applicables telles quelles dans les grands groupes. De nouvelles problématiques de cohésions globales doivent être...



MATTHIEU FRONTON
LE GROUPE LA POSTE

MA VIE DE RSSI DANS LE MONDE DEVOPS

Je vous propose de partager mon expérience en tant que RSSI, sur la découverte et sur l'appréhension de cette nouvelle façon de réaliser des projets.



BENOÎT FUZEAU
CASDEN BANQUE POPULAIRE & REPRÉSENTANT DU CLUSIF

TALES OF CONTAINER SECURITY - OU L'IMPORTANCE D'UNE...

Les conteneurs sont partout, laptop, serveurs, cloud hosting, embarqué, edge computing, ils prennent littéralement le monde d'assaut ! Cette tempête s'accompagne d'un certain nombre de...



RACHID ZAROUALI
SEVENSHERE

... et beaucoup d'autres ...

Sources :

Inténe Orange

<https://twitter.com/jvehent/status/1230574950769184771>

<https://2019.devopsrex.fr/>

1. Pourquoi ce cours ?

Les fonctionnements en mode DevOps et agile deviennent la norme.

Le DevOps et l'agilité changent fortement l'approche sécurité.

Une absence de prise en compte de la sécurité dans un « projet DevOps » fait prendre des risques importants :

- L'automatisation qui fait passer à l'échelle les bonnes (et mauvaises) pratiques.
- Moins de filtrage réseaux, exposition plus large.
- Des livraisons de features très régulières : l'audit traditionnel n'est plus suffisant.

2. Introduction à l'agilité et au DevOps

2. Introduction à l'agilité et au DevOps

Au delà du buzzword

DevOps et agile sont "tendances" et utilisés pour tout et n'importe quoi.

Ce n'est pas aisément de décrire ces évolutions qui ne sont pas que techniques mais davantage culturelles et organisationnelles.

Les grandes entreprises ont pris conscience de l'implication entre "performance IT" et "performance business" notamment par la démonstration des GAFA et des startups.

Les grandes entreprises recopient et adaptent ces modèles, c'est la "transformation numérique" dans de nombreux secteurs.

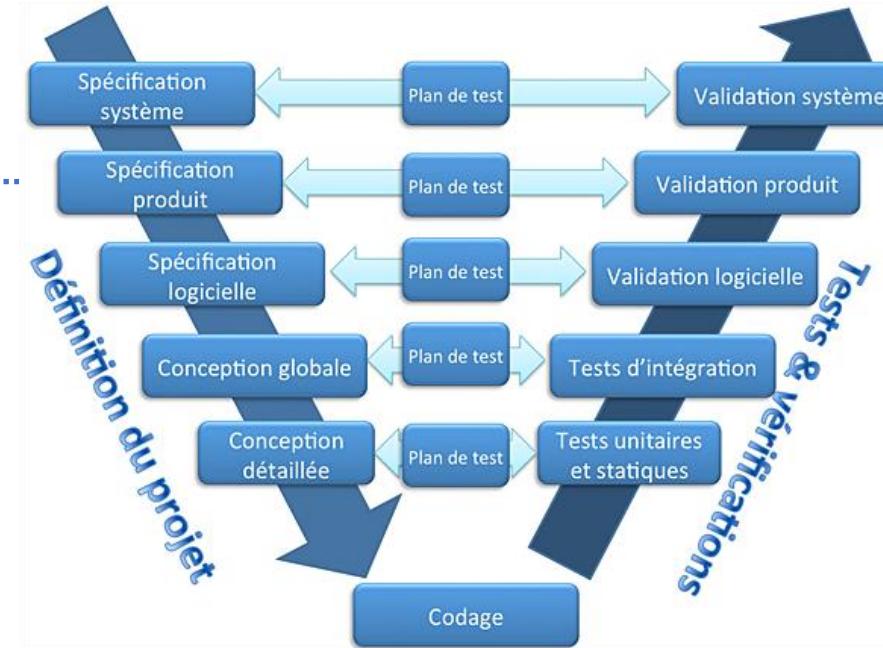
2. Introduction à l'agilité et au DevOps

Au delà du buzzword

Le mode de fonctionnement historique : le cycle en V

Pourquoi doit-on changer de modes de fonctionnement ?

- Un monde VUCA
 - Volatility
 - Uncertainty
 - Complexity
 - Ambiguity
- Une concurrence accrue qui nécessite de répondre le plus rapidement et « le mieux possible » aux besoins du client, pour créer un avantage concurrentiel
 - Que veut le client ?
 - Est-il satisfait ?
 - Quelle qualité ?



2. Introduction à l'agilité et au DevOps

Agile : Manifeste agile

Manifeste agile 2001

Défini 4 grands principes et 12 principes sous-jacents.



2. Introduction à l'agilité et au DevOps

Agile : les 12 principes sous-jacents

#1 Take an economic view

#2 Apply systems thinking

#3 Assume variability; preserve options

#4 Build incrementally with fast, integrated learning cycles

#5 Base milestones on objective evaluation of working systems

#6 Visualize and limit WIP, reduce batch sizes, and manage queue lengths

#7 Apply cadence, synchronize with cross-domain planning

#8 Unlock the intrinsic motivation of knowledge workers

#9 Decentralize decision-making

#10 Organize around value

© Scaled Agile, Inc.

2. Introduction à l'agilité et au DevOps

Agile : les 12 principes sous-jacents



- #1 Take an economic view
- #2 Apply systems thinking
- #3 Assume variability; preserve options
- #4 Build incrementally with fast, integrated learning cycles
- #5 Base milestones on objective evaluation of working systems
- #6 Visualize and limit WIP, reduce batch sizes, and manage queue lengths
- #7 Apply cadence, synchronize with cross-domain planning
- #8 Unlock the intrinsic motivation of knowledge workers
- #9 Decentralize decision-making
- #10 Organize around value

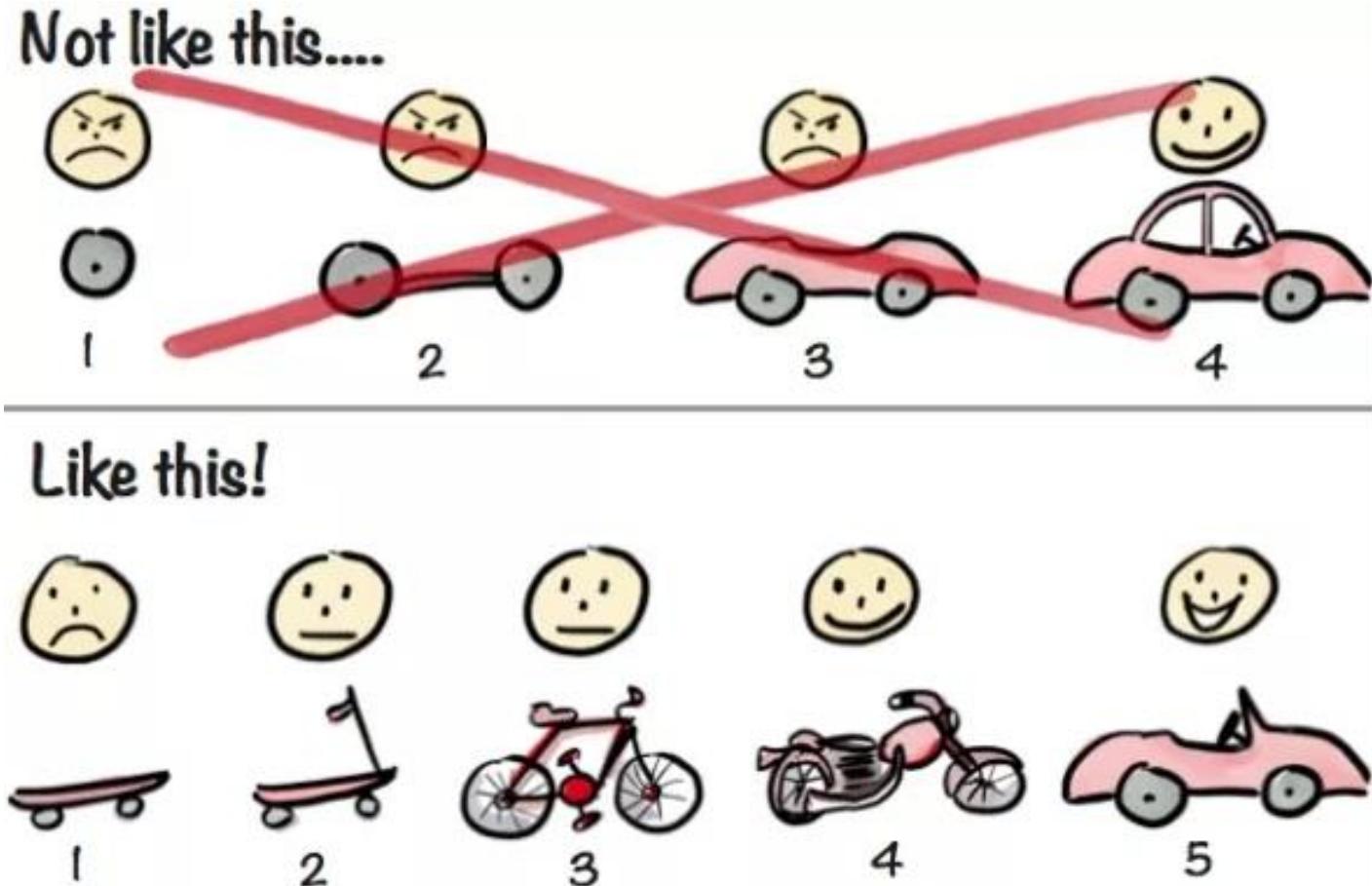
© Scaled Agile, Inc.

2. Introduction à l'agilité et au DevOps

« *Build incrementally with fast integrated learning cycles* » : Agilité versus cycle en V

Débute par un MVP (Minimal Viable Product) sur lequel on a déjà un retour du client.

Puis ajout par incrément successifs des nouvelles fonctionnalités avec **retour client permanent** (idéalement, il faut pouvoir le mesurer).



2. Introduction à l'agilité et au DevOps

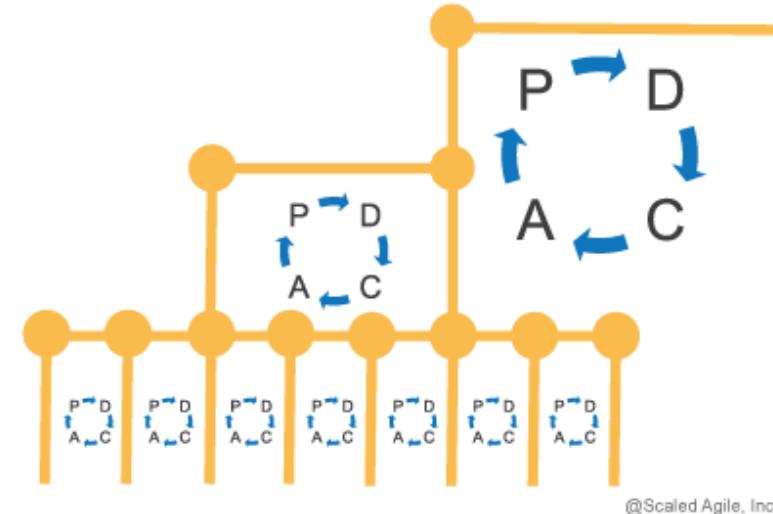
« ***Build incrementally with fast integrated learning cycles*** » : l'amélioration continue

On se base sur les retours du terrain (le client)

- Demande de retours au client (questionnaire de satisfaction, sondages)
- Retours spontanés du client (appel en service client, demande d'aide, signalisation de bug)
- Mesures dérivées (évolution du nombre d'actes réussis, adoption d'une fonctionnalité)

Pour mettre en place l'amélioration continue

- PDCA : Plan / Do / Check / Act



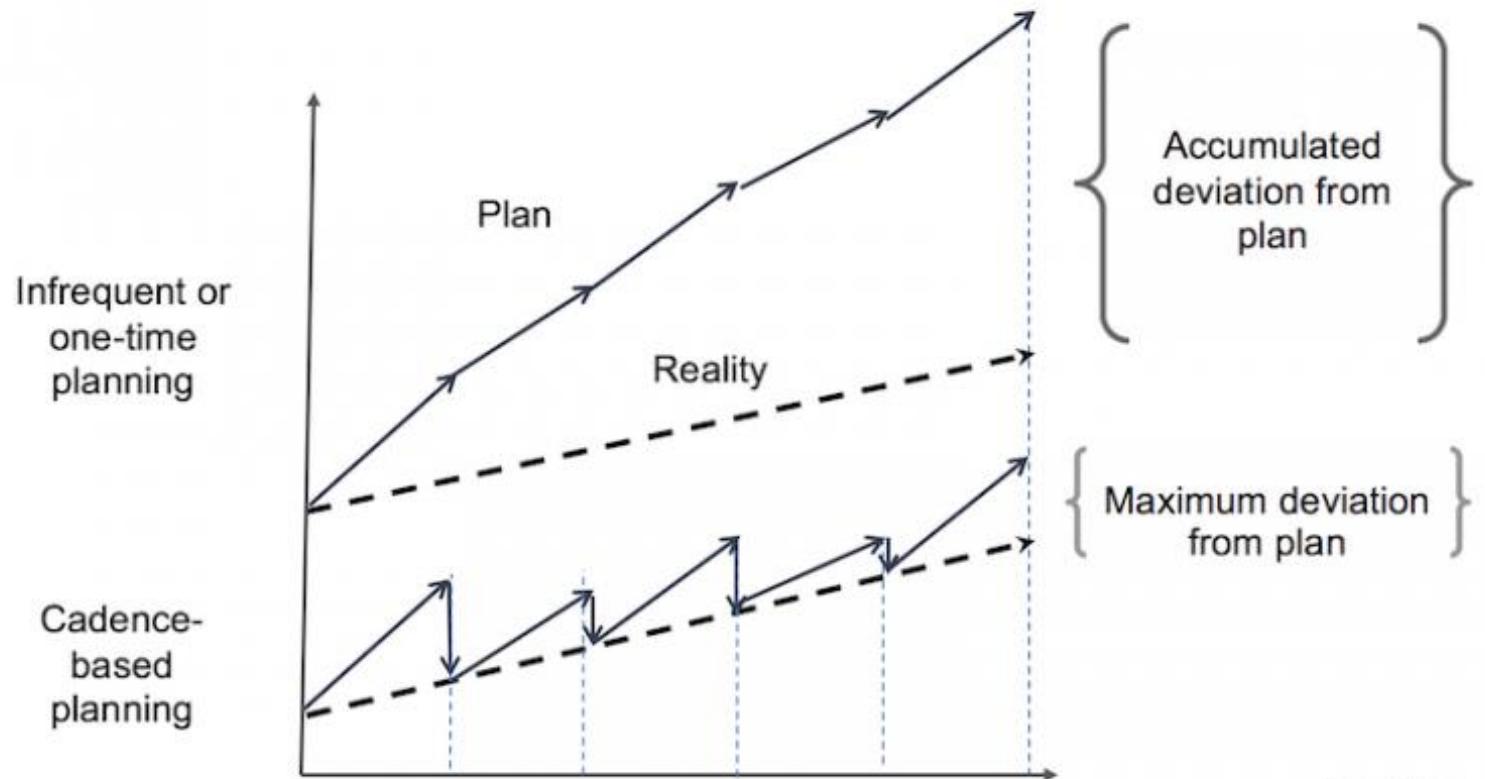
A la maille : de l'équipe / d'un ensemble d'équipes / de l'entreprise

2. Introduction à l'agilité et au DevOps

« *Build incrementally with fast integrated learning cycles* » : réajuster en continu

Cycle en V

Agilité



© Scaled Agile, Inc.

2. Introduction à l'agilité et au DevOps

Agile : les 12 principes sous-jacents

- 
- #1 Take an economic view
 - #2 Apply systems thinking
 - #3 Assume variability; preserve options
 - #4 Build incrementally with fast, integrated learning cycles
 - #5 Base milestones on objective evaluation of working systems
 - #6 Visualize and limit WIP, reduce batch sizes, and manage queue lengths
 - #7 Apply cadence, synchronize with cross-domain planning
 - #8 Unlock the intrinsic motivation of knowledge workers
 - #9 Decentralize decision-making
 - #10 Organize around value

© Scaled Agile, Inc.

2. Introduction à l'agilité et au DevOps

« Visualize & limit WIP »

Basé sur les travaux relatifs à l'efficacité

- files d'attente, « just in time manufacturing» de Toyota, etc)

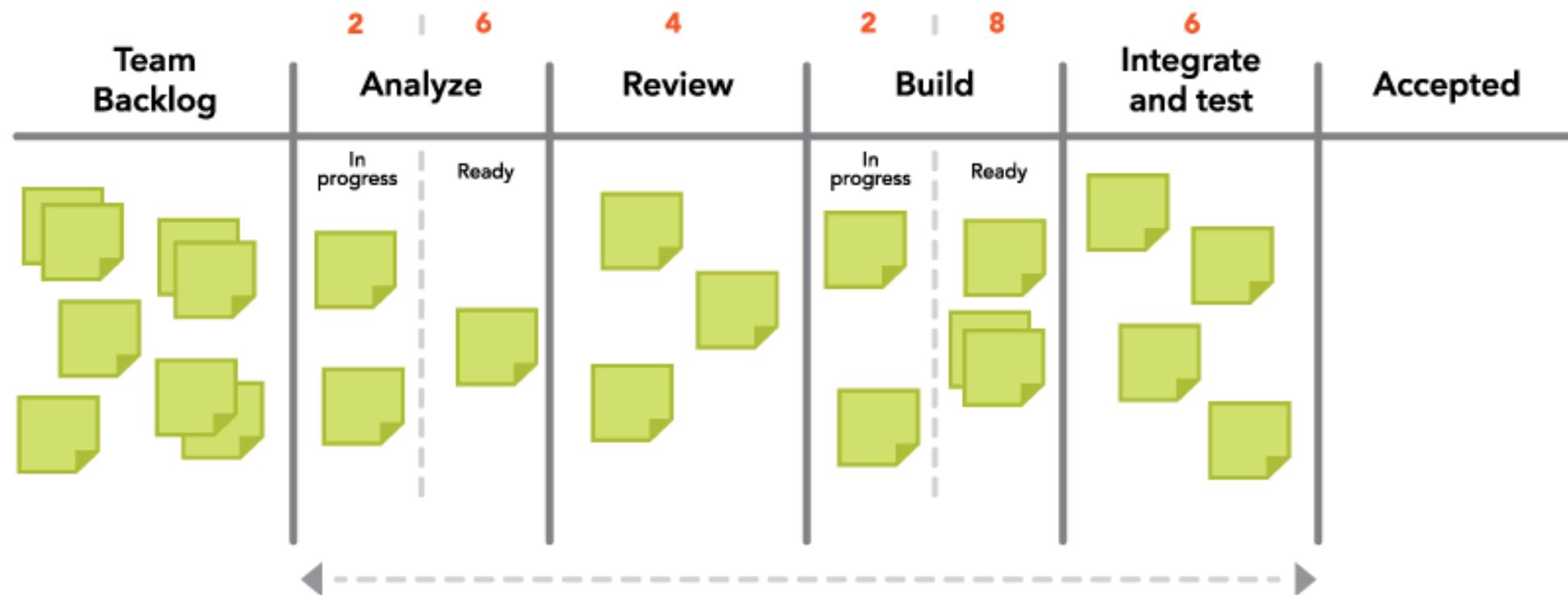
Applicable à tout type de projet (cf « personal Kanban »)

Le principe :

- Diviser le travail à accomplir en tâches de complexité faible pour améliorer la prédictibilité
- Limiter le Work In Progress pour augmenter le débit

2. Introduction à l'agilité et au DevOps

« Visualize & limit WIP »



© Scaled Agile, Inc.

2. Introduction à l'agilité et au DevOps

Agile : les 12 principes sous-jacents

#1 Take an economic view

#2 Apply systems thinking

#3 Assume variability; preserve options

#4 Build incrementally with fast, integrated learning cycles

#5 Base milestones on objective evaluation of working systems

#6 Visualize and limit WIP, reduce batch sizes, and manage queue lengths

#7 Apply cadence, synchronize with cross-domain planning

#8 Unlock the intrinsic motivation of knowledge workers

#9 Decentralize decision-making

#10 Organize around value

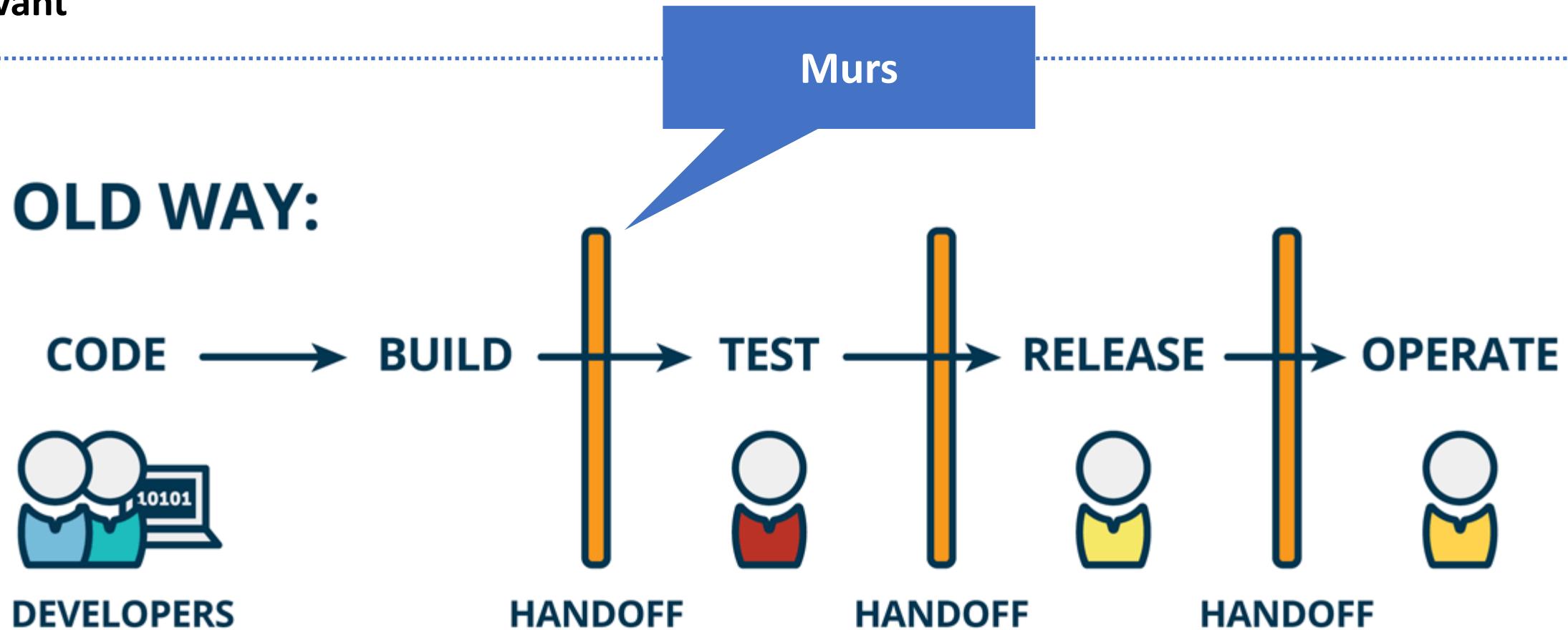


© Scaled Agile, Inc.

2. Introduction à l'agilité et au DevOps

Avant

OLD WAY:

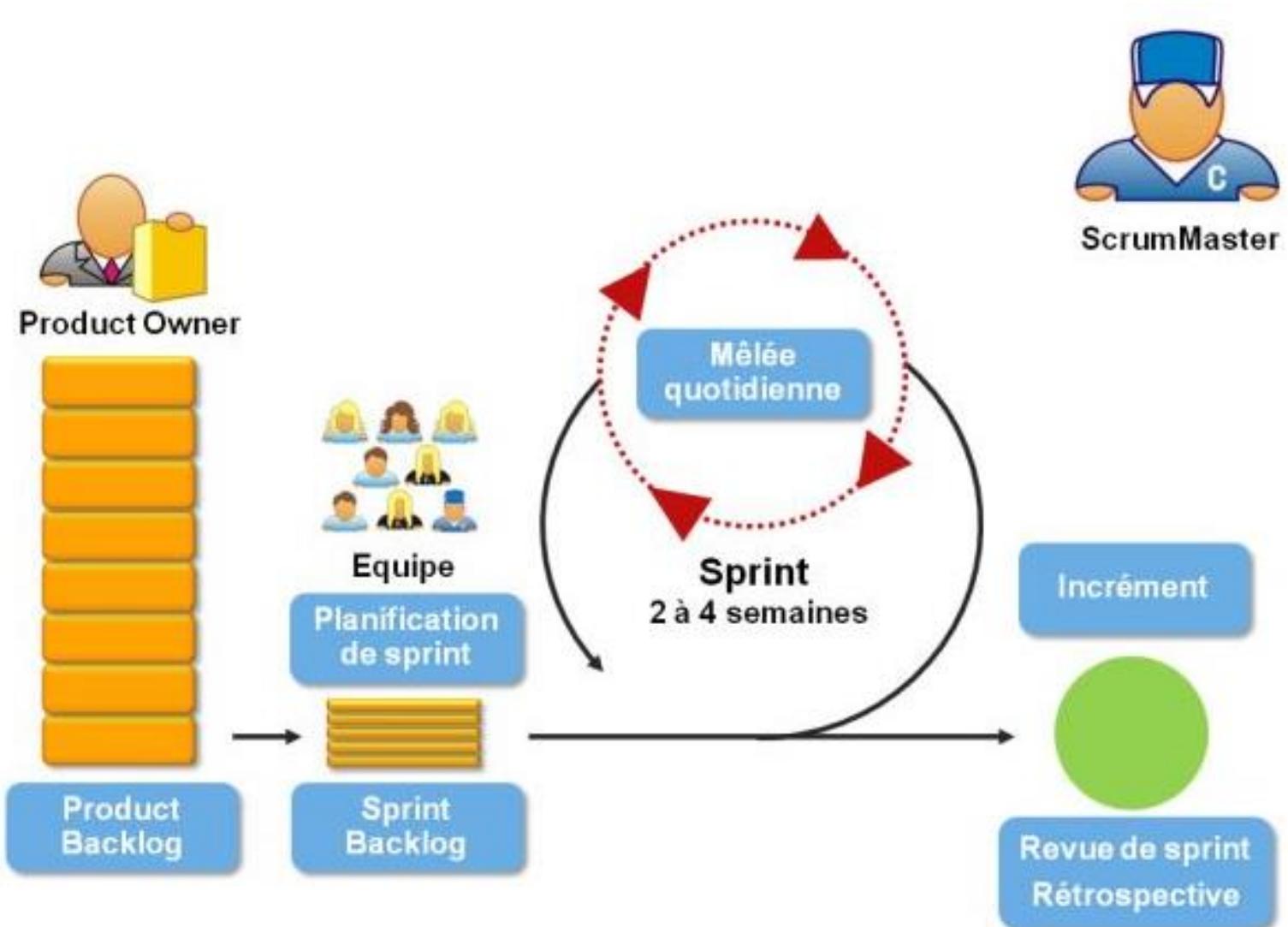


2. Introduction à l'agilité et au DevOps

Agile : Scrum

Méthode Scrum par sprints.

En opposition au mode historique : cycle en V.



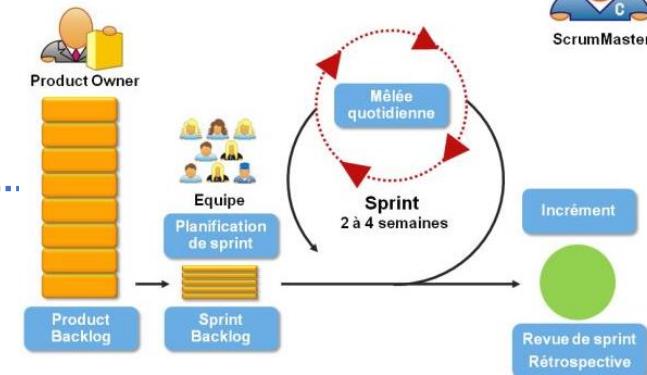
2. Introduction à l'agilité et au DevOps



Agile : Scrum

Le Product Owner (PO)

- Définit ce qui doit être fait (= « les stories »)
- Priorise les stories
- Porte la voix du client



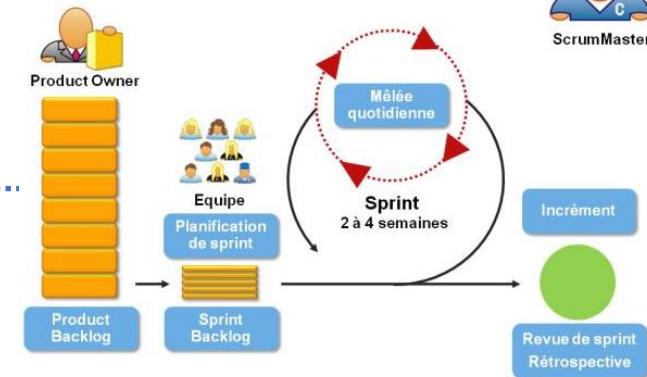
2. Introduction à l'agilité et au DevOps



Agile : Scrum

Le Scrum Master

- Facilitateur du travail de l'équipe
- S'assure que l'équipe a tout ce qu'il faut pour avancer
- Remonte les difficultés au management / au PO
- Préparer et coordonne les « cérémonies agiles »



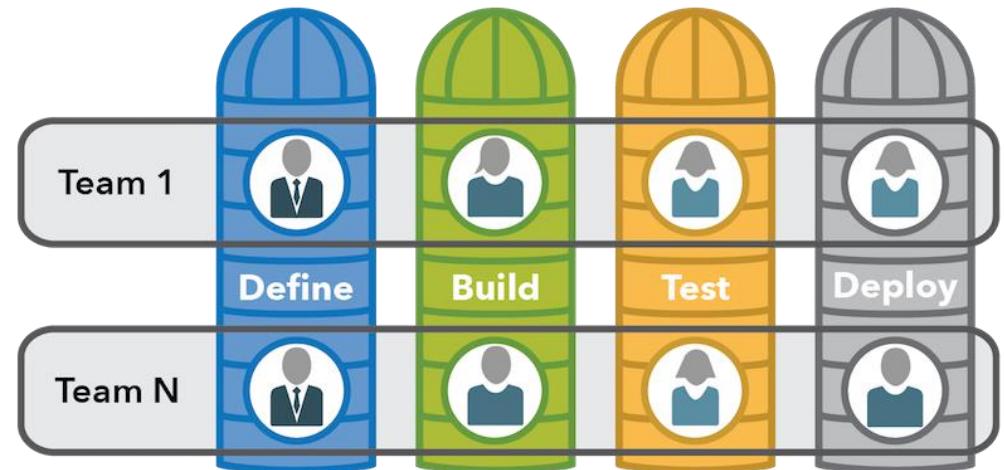
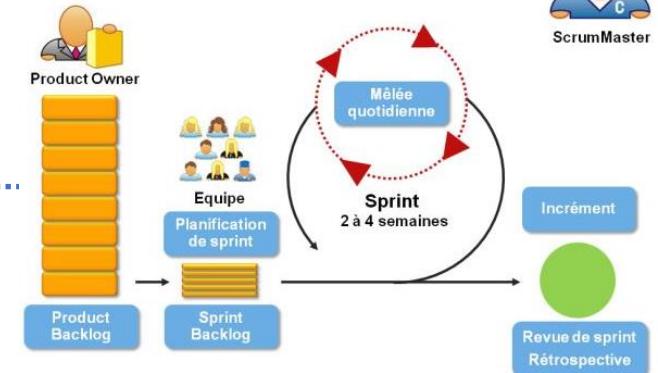
2. Introduction à l'agilité et au DevOps



Agile : Scrum

L'équipe

- Des Devs, des Ops, des testeurs
- De 3 à 9 personnes en théorie...
- Une entité auto-organisée (indépendance)
- Une entité cross-fonctionnelle
- Implémente les fonctionnalités attendues
- Délivre de la valeur sur des itérations courtes



© Scaled Agile, Inc.

2. Introduction à l'agilité et au DevOps

DevOps



Principes du DevOps :

1. Améliorer la **performance de l'ensemble de la chaîne** (pas juste avoir des super Dev ou juste des super Ops) pour fournir le meilleur produit possible.
2. Mettre en place une **boucle de feedback** rapide pour ajuster la chaîne. (feedback des Ops, feedback des clients).
3. Créer une **culture de l'expérimentation et de l'apprentissage des erreurs** afin d'améliorer la résilience et confiance dans l'ensemble du système.

Très bien, comment peut-on faire ? → les piliers "CALMS" donnent des pistes.

2. Introduction à l'agilité et au DevOps



Les piliers CALMS

Culture	Entraide Dev + Ops, feedback en continu, motivation pour l'expérimentation
Automation	Automatiser, rendre déterministe, passer à l'échelle
Lean	Amélioration continue, optimisation du fonctionnement
Measure	Définir des indicateurs, prendre les décisions sur ces indicateurs
Share	Partage dans l'équipe et entre les équipes, capitalisation des connaissances

2. Introduction à l'agilité et au DevOps

Concrètement ?

Côté technique, en exemples concrets (et non-exhaustifs), le DevOps peut impliquer :

- Une chaîne d'intégration continue.
- L'automatisation des déploiements.
- L'automatisation des tests.
- Des rétrospectives sur ce qui ne marche pas et comment le corriger.

2. Introduction à l'agilité et au DevOps

La sécurité DevOps

Pourquoi une sécurité DevOps ? (et pas l'approche sécurité historique)

Les changements induits par le DevOps et l'agilité :

- organisation différente,
- technologies différentes,

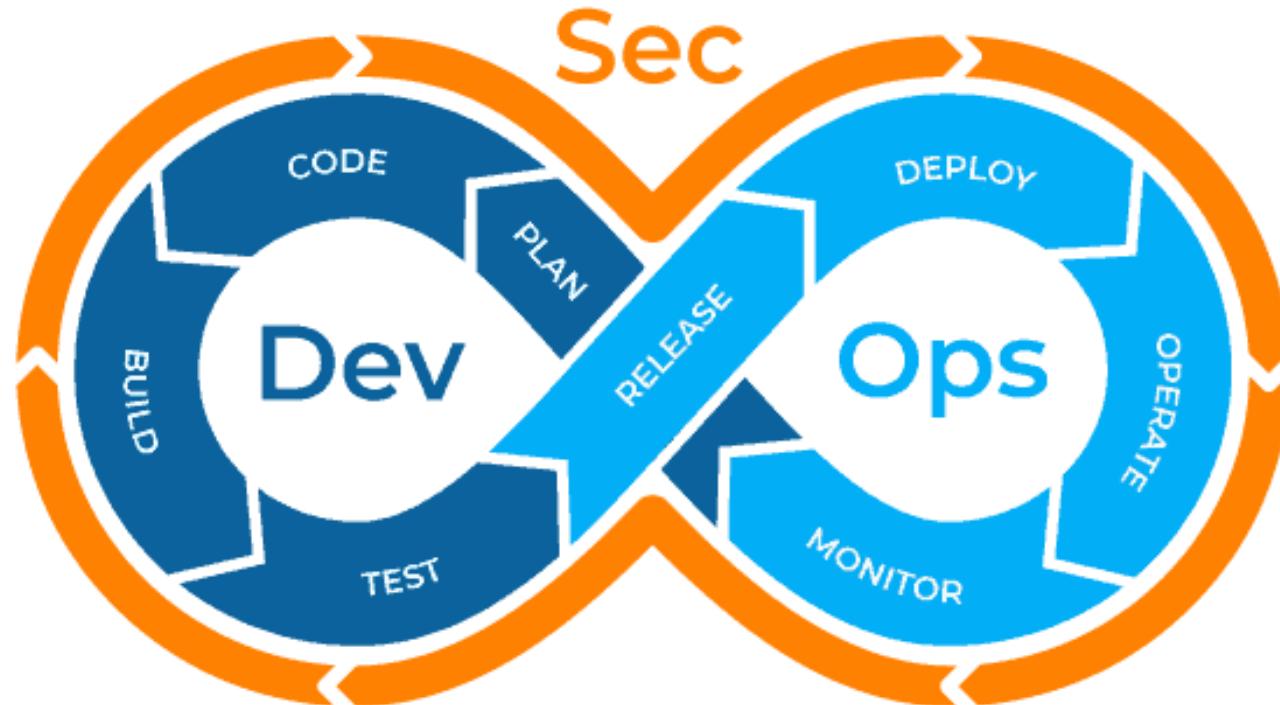
obligent la sécurité à devoir évoluer pour rester efficace et :

- ne pas **se retrouver exclus** du cycle de vie IT.
- ne pas avoir une **illusion de contrôle** alors que "les choses" se passent deux couches de virtualisations au dessus.

2. Introduction à l'agilité et au DevOps

Le buzzword

Le buzzword tendance : DevSecOps



2. Introduction à l'agilité et au DevOps

Ce que le DevOps n'est pas

Le DevOps **n'est pas** une technologie.

Le DevOps **n'est pas** un outil.

Le DevOps **ne devrait pas** être la fonction d'une personne.

Le DevOps **ne devrait pas** être une équipe dans l'entreprise.

Le DevOps **ne devrait pas** être le domaine de l'exploitation informatique.

Plan du cours

C'est un très vaste sujet, qui reprend de nombreux aspects de la sécurité "habituelle" et qui en apporte de nouveaux aspects :

2. Introduction à l'agilité et au DevOps
3. Sécurité dans l'**intégration continue**
4. Tests de sécurité par analyse statique (**SAST**)
5. Tests de sécurité par analyse dynamique (**DAST**)
6. Tests de sécurité interactifs (**IAST**)
7. Protection du contexte d'exécution (**RASP**)
8. Sécurité de l'**Infrastructure as Code**
9. Sécurité des **containers**
10. Gestion des **secrets**

3. Intégration continue

3. Intégration continue

Dans le monde historique (sans intégration continue)

Les Devs (ou équipes de Dev) développent chacun leur partie de logiciel, en silos.

Puis, tous les 1 / 2 / n mois, un assemblage (intégration) des différentes parties est faite tentée.

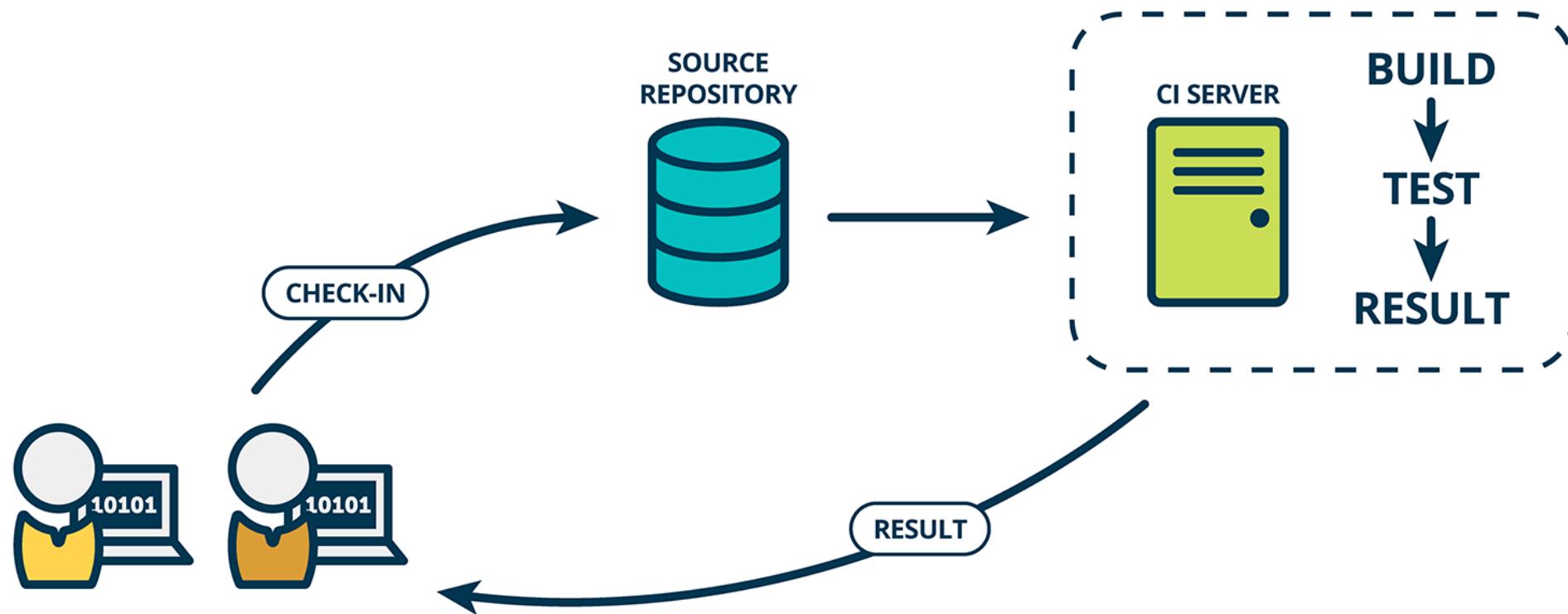
L'intégration mène à :

- des problèmes de fusion qui prennent beaucoup de temps à corriger → \$\$
- du code à adapter (nouveaux bugs / *rework*) → \$\$
- les Dev qui ne sont plus disponibles pour de nouvelles évolutions durant la période d'intégration → \$\$

En pensant gagner du temps (de l'argent), on en perd sur le long terme.

3. Intégration continue

Avec intégration continue



3. Intégration continue

Avantages de l'intégration continue

Avantages fonctionnels

- Limite les problèmes de fusion de code.
- Donne un feedback quasi immédiat aux Dev pour une correction à moindre coût (plutôt qu'une difficulté à gérer plusieurs mois après).
- Améliore la qualité et rapidité par intégration de tests automatisés.

Avantages sécurité

- Une meilleure qualité de code : c'est toujours bon à prendre côté la sécurité.
- Mise en place de tests sécurité systématiques (audit statique, analyse des dépendances, etc.).
- Correction moins couteuse des bugs de sécurité.

3. Intégration continue

Quelques solutions d'intégration continue



Jenkins



GitLab CI



Travis CI

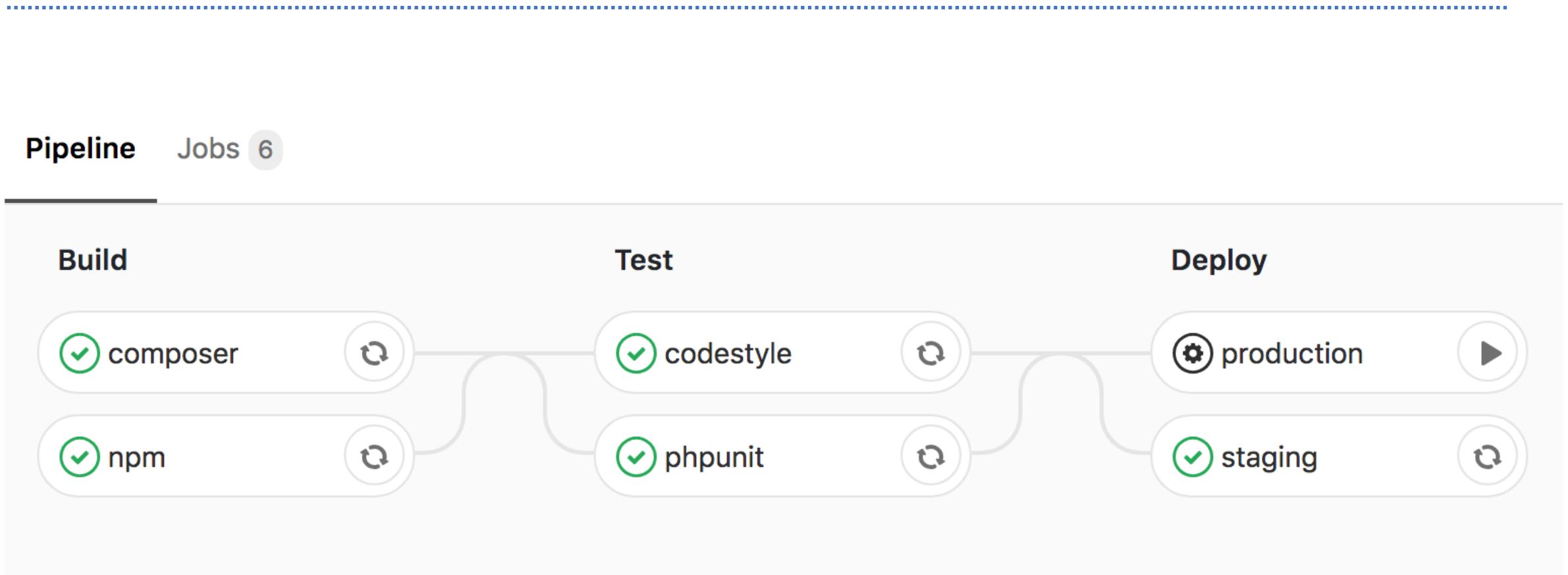


On premise

Cloud SaaS

3. Intégration continue

Exemples visuels



passed

Build #402 for commit [36f18e7f](#) from [feature/test-runner-12905](#) by Thomas Gelf about a minute ago

```
Running with gitlab-ci-multi-runner 1.7.1 (f896af7)
Using Shell executor...
Running on gitlab-jessie1...
Fetching changes...
HEAD is now at 8e2bb8b Update .gitlab-ci.yml
From http://gitlab1.lxd/Icinga/icingaweb2-module-director
  8e2bb8b..36f18e7 feature/test-runner-12905 -> origin/feature/test-runner-12905
Checking out 36f18e7f as feature/test-runner-12905...
$ mysql -u root -e "CREATE DATABASE $DIRECTOR_TESTDB"
$ phpunit --testdox --colors
PHPUnit 4.2.6 by Sebastian Bergmann.

Configuration read from /home/gitlab-runner/builds/90082195/0/Icinga/icingaweb2-module-director/phpunit.xml

s\Icinga\Module\Director\CustomVariable\CustomVariables
[x] Whether special key names
[x] Vars can be unset and set again
[x] Variables to expression

s\Icinga\Module\Director\IcingaConfig\AssignRenderer
[x] Whether equal match is correctly rendered
[x] Whether wildcards render a match method
[x] Whether a combined filter renders correctly

s\Icinga\Module\Director\IcingaConfig\ExtensibleSet
[x] No values result in empty set
[x] Values passed to constructor are accepted
[x] Constructor accepts single values
[x] Single values can be blacklisted
[x] Multiple values can be blacklisted
[x] Simple inheritance works fine
[x] We can inherit from multiple parents
[x] Own values override parents
[x] Inherited values can be blacklisted
[x] Inherited values can be extended
```

Build details

Duration: 3 seconds

Finished: about a minute ago

Runner: #10

[Raw](#)

[Erase](#)

Commit title

Update .gitlab-ci.yml

Tags

[director](#) [jessie](#)

Xenial/PostgreSQL

Xenial/MySQL

Jessie/PostgreSQL

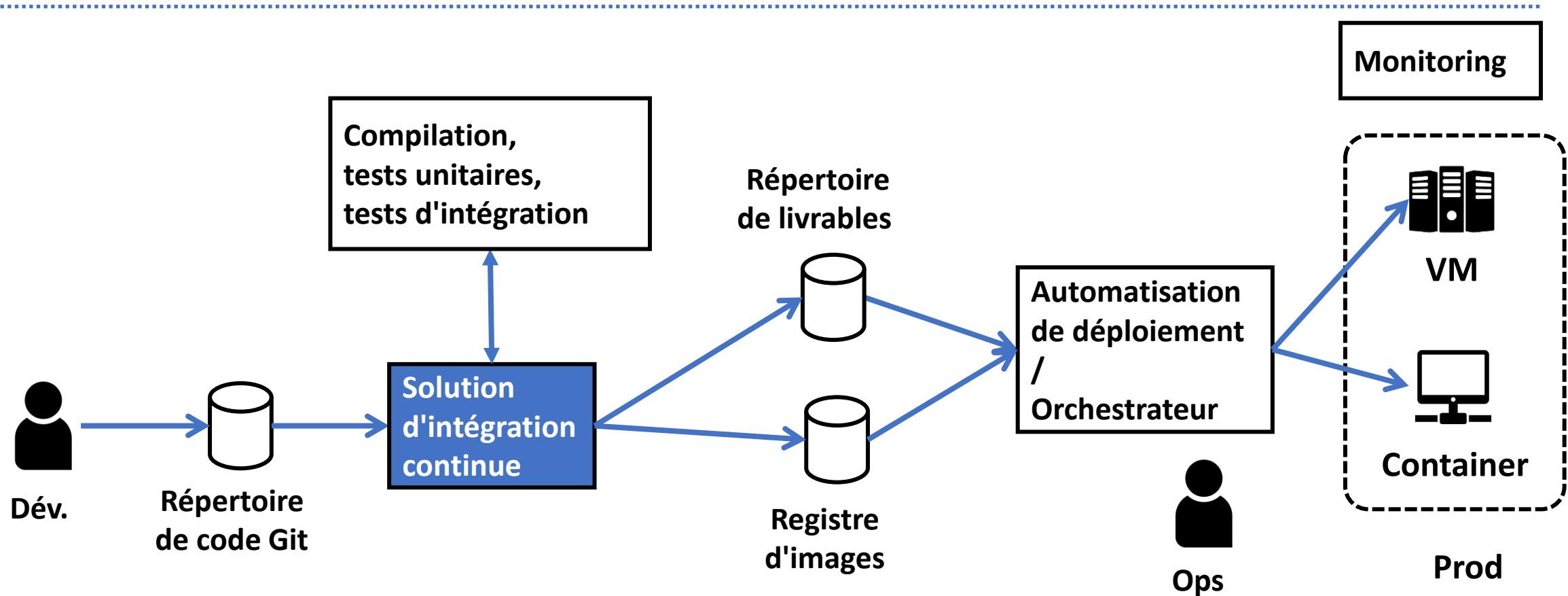
CentOS 7/PostgreSQL

Jessie/MySQL

CentOS 7/MySQL

3. Intégration continue

Schématisation



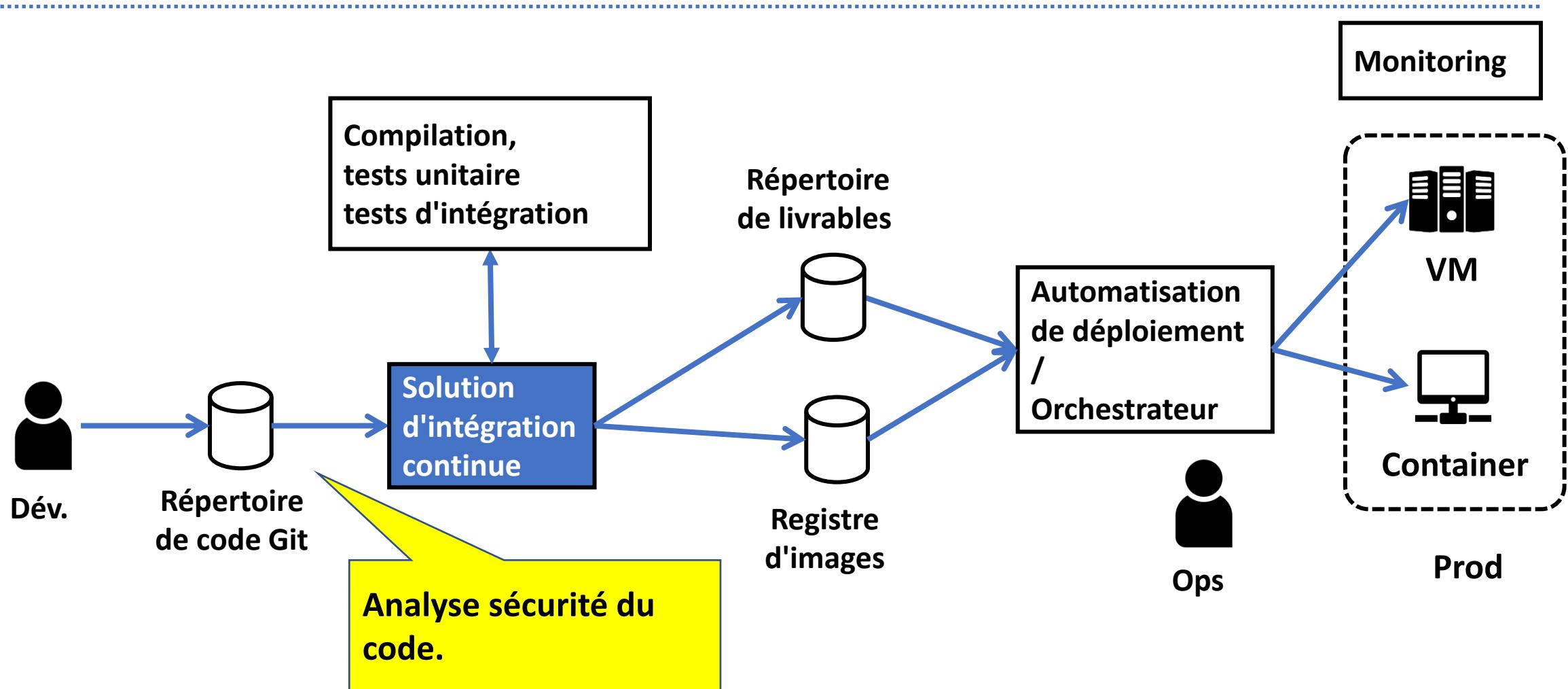
3. Intégration continue

Opportunités pour la sécurité

La sécurité peut doit tirer partie de l'intégration continue (et sa capacité à tester) pour y placer des tests de sécurité.

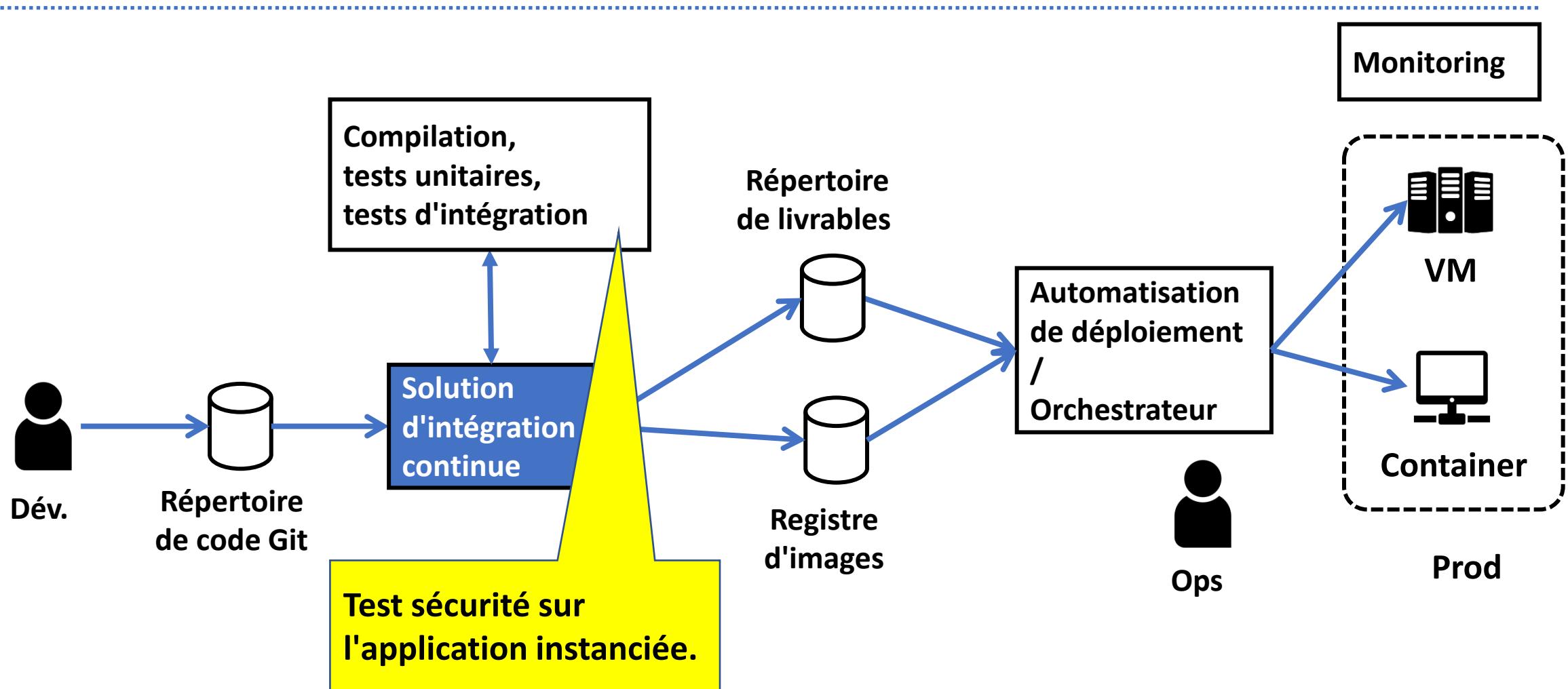
3. Intégration continue

Tests de sécurité



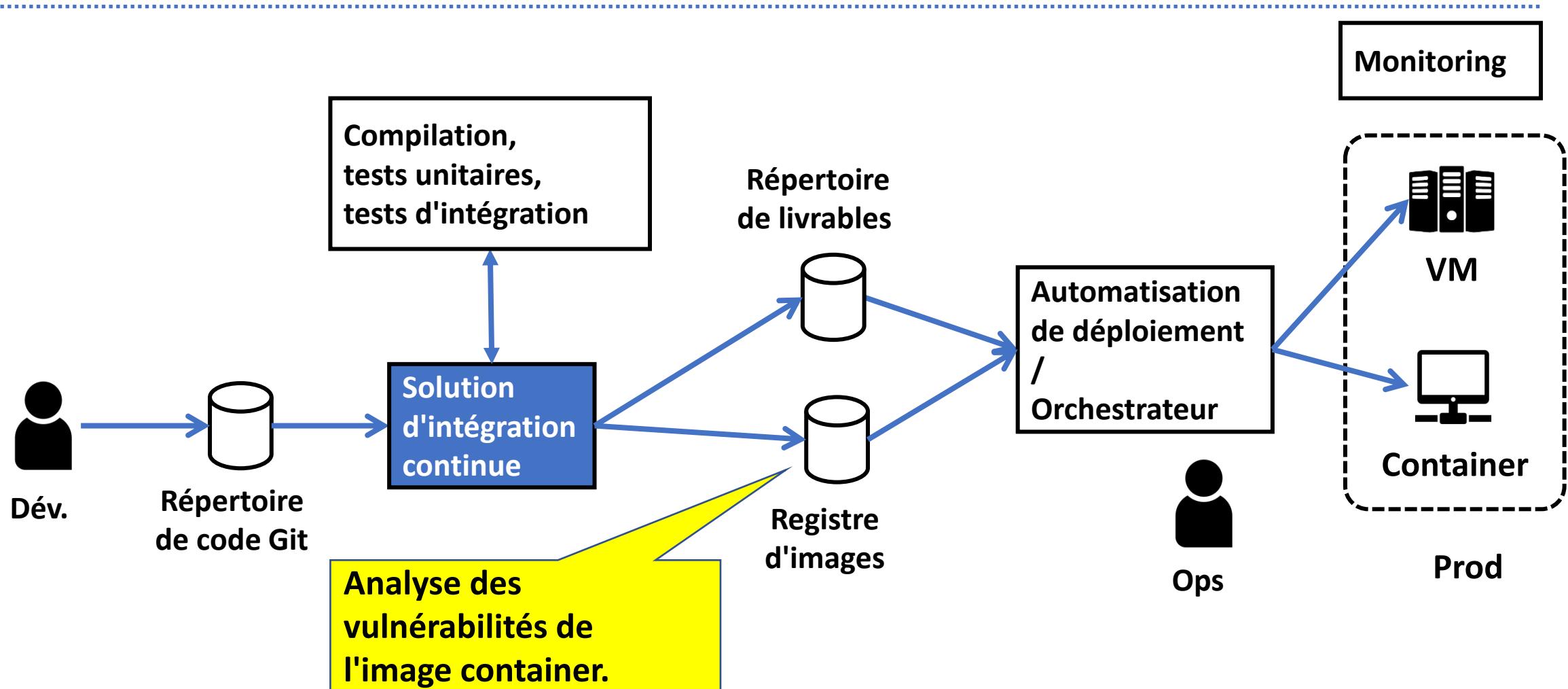
3. Intégration continue

Tests de sécurité



3. Intégration continue

Tests de sécurité



3. Intégration continue

Quelques risques et vigilances sécurité

En contrepartie des nouvelles opportunités d'intégration de la sécurité dans l'intégration continue, de nouvelles vigilances.

3. Intégration continue

Quelques risques et vigilances sécurité

Vigilance sur les solutions de CI "on premise" :

- Quels credentials la CI utilise pour récupérer du code sur le dépôt ?
- L'IHM de la CI cloisonne-t-elle l'accès aux différents projets gérés ?
- Comment gère-t-on les accès à la CI pour les Dev de différents produits ?
- A-t-on confiance dans les machines "runners" qui exécutent les tests ?
- La CI est-t-elle "root" sur un ensemble de VM pour déployer le code ?
- La CI est-t-elle "admin" sur un cloud pour déployer du code / VM ?

3. Intégration continue

Quelques risques et vigilances sécurité

Vigilance sur les solutions de CI "cloud SaaS" :

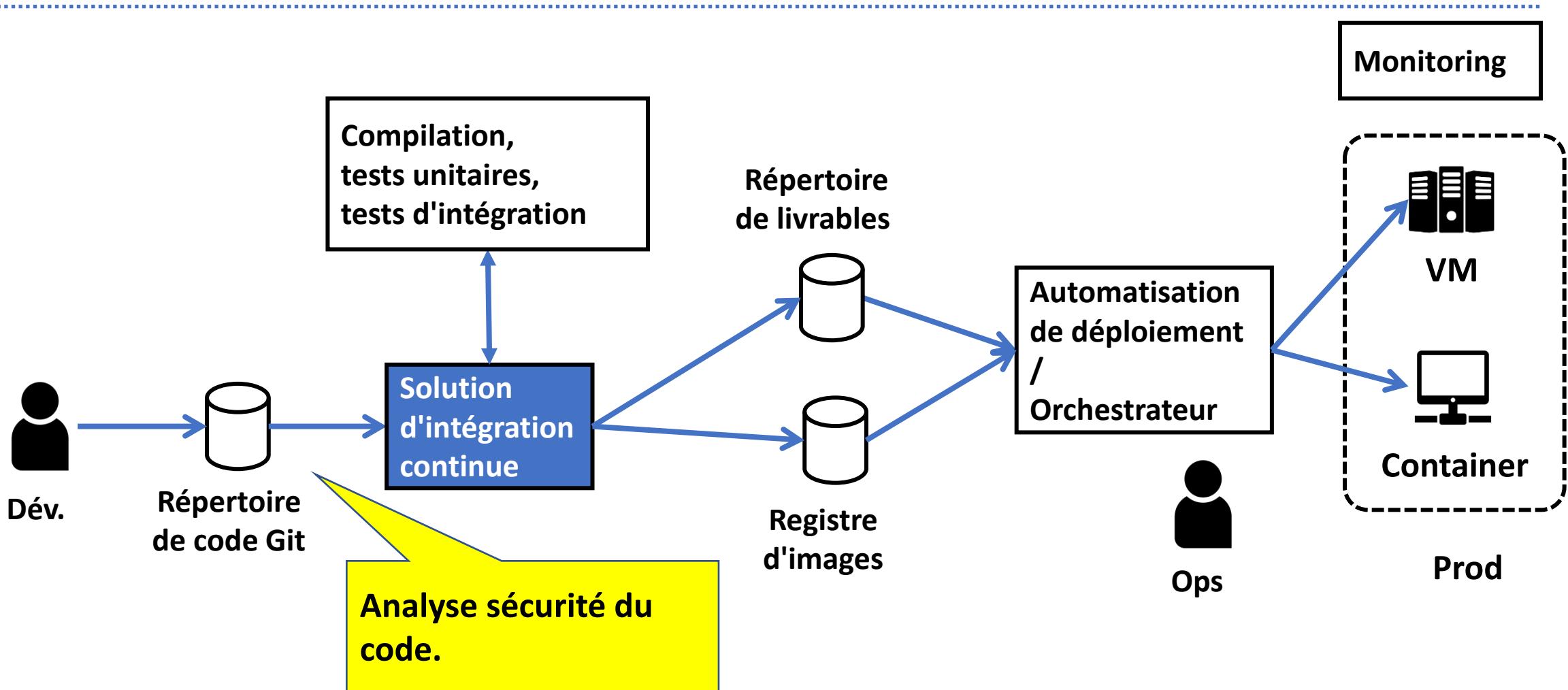
Mêmes questions pour une CI on premise, et en plus :

- A-t-on confiance dans le fournisseur cloud ?
- A-t-on renforcé l'authentification du fait que l'IHM est exposée sur Internet ?
(→ 2FA)

4. SAST – Analyse statique

4. SAST – Analyse statique

Tests de sécurité



4. SAST – Analyse statique

Introduction à SAST

Static Application Security Testing

Par "statique", on entend : le code pur, ni compilé, ni exécuté.

On analyse le code par un outil automatisé.

C'est donc une approche « boîte blanche ».

4. SAST – Analyse statique

Différents types de tests

Différentes possibilités de test que l'on doit combiner :

- Analyse de dépendances
- Recherche de secrets en clair dans le code
- Lint
- Lint (orienté sécurité)
- Recherche de vulnérabilités

4. SAST – Analyse statique

Analyse de dépendances

Les projets intègrent de plus en plus (dizaines voire centaines) de dépendances open-source, souvent des bibliothèques de code.

Le problème : comment identifier quand une dépendance d'un projet est vulnérable ? Nécessaire pour mettre à jour la dépendance vers une version corrective.

L'approche : Listing des dépendances (*libraries*) logicielles d'une application.
Comparaison de cette liste avec des référentiels de vulnérabilités en ligne.

4. SAST – Analyse statique

Exemple d'outil d'analyse de dépendances

OWASP Dependency-Check (surtout pour Java et .Net)

NodeJS : npm audit

4. SAST – Analyse statique

Analyse de dépendances

Identifier les vulnérabilités dans les dépendances, c'est déjà très bien.

Ensuite il faut avoir la culture :

- Comment patche-t-on ?
- A-t-on anticipé l'impact d'une montée de version d'une dépendance sur le reste du code ?

Status	Component	Version
✗ 1 vulnerability	ansi-gray	3.1.7
✗ 1 vulnerability	ansi-styles	6.3.11
MEDIUM	Insecure variable usage	
✓ Safe	archy	14.14.7
✓ Safe	arr-union	13.11.12
✗ 3 vulnerabilities	array-slice	7.1.0
✗ 4 vulnerabilities	beeper	1.9.4

4. SAST – Analyse statique

Recherche de secrets dans le code

Un exemple : **Gitleaks**

Recherche de secret en clair en passant un repo Git en entrée.

Peut être utilisé aussi bien :

- En ligne de commande (lancement manuel)
- Dans une chaîne d'intégration continue (lancement automatiquement)
- En *hook de pre commit* (lancement local avant commit effectif)

4. SAST – Analyse statique

```
$ gitLeaks --json https://github.com/zricethezav/gronit
Cloning https://github.com/zricethezav/gronit...
{
  "line": "+const AWS_KEY = \"AKIALALEMEL332430LIAE\"",
  "commit": "cb5599aeed261b2c038aa4729e2d53ca050a4988",
  "string": "AKIALALEMEL332430LIA",
  "reason": "AWS",
  "commitMsg": "fake key",
  "time": "2018-02-04 19:10:58 -0600",
  "author": "Zachary Rice",
  "file": "main.go",
  "repoURL": "https://github.com/zricethezav/gronit"
}
{
  "line": "-const AWS_KEY = \"AKIALALEMEL332430LIAE\"",
  "commit": "eaefffdc65b4c73ccb67e75d96bd8743be2c85973",
  "string": "AKIALALEMEL332430LIA",
  "reason": "AWS",
  "commitMsg": "remove fake key",
  "time": "2018-02-04 19:43:28 -0600",
  "author": "Zachary Rice",
  "file": "main.go",
  "repoURL": "https://github.com/zricethezav/gronit"
}
Report written to /Users/Zach/.gitLeaks/report/zricethezav/gronit_leaks.json
```

4. SAST – Analyse statique

Lint

Le linter peut identifier :

- Des erreurs de programmation (avant de passer au compilateur / interpréteur).
- Des erreurs stylistiques.
- Des constructions suspectes ou ambiguës.
- L'utilisation de fonctions qui sont / vont devenir dépréciées.
- L'utilisation de fonctions dangereuses.

Les linters aussi peuvent être intégrés en amont dans les IDE de développement (ce qui évitera les erreurs avant même l'envoi vers le linter de la CI).

4. SAST – Analyse statique

Des exemples de linters

Qualité de code : SonarQube

Python : pylint / Flake8

JavaScript / NodeJS : ESLint

YAML : YAMLLing

JSON : JSONLint

Ansible : Molecule

Dockerfile : Hadolint

4. SAST – Analyse statique

Lint orienté sécurité

En extension des linters pour la qualité du code, des linters sécurité ont vu le jour.

Ils ne réalisent souvent que des recherche par expressions régulières :

« Attention, cette fonction exec() ne devrait pas être utilisée ».

4. SAST – Analyse statique

Des exemples de linters orientés sécurité

Python : bandit (et linter non sécu : pylint / Flake8)

JavaScript : ESLint Security plugin

Java : Find Security Bugs

NodeJS : NodeJsScan

PHP : phpcs-security-audit

Go : Gosec

Projets par toujours bien maintenus. À vérifier et tester.

4. SAST – Analyse statique

Recherche de vulnérabilités

Les outils SAST de recherche de vulnérabilités analysent le code plus en profondeur que les linters sécurité.

Pour identifier :

- Les vulnérabilités type Top 10 OWASP (XSS, SQLI, CSRF, ...).
- Erreur d'utilisation des fonctions cryptographiques.
- Erreur de configuration.
- Utilisation de fonctions dangereuses.

4. SAST – Analyse statique

Analyse de vulnérabilités

Exemples d'outils de recherches de vulnérabilités

- Checkmarx
- SonarQube Enterprise (ou version Community avec plugins de sécurité)
- Fortify
- Coverity

Ils sont tous plus ou moins spécialisés dans une liste de langages.
L'outil parfait n'existe pas (encore).

4. SAST – Analyse statique

Analyse de vulnérabilités

Exemple de résultat

Request to merge [awesome-feature](#) into [master](#) [Check out branch](#)

Pipeline #18777035 passed for 8805f6cd.

SAST improved on 1 security vulnerability and degraded on 4 security vulnerabilities [Collapse](#)

- Medium: Cipher with no integrity in [src/main/java/com/gitlab/security_products/tests/App.java:29](#)
- Medium: ECB mode is insecure in [src/main/java/com/gitlab/security_products/tests/App.java:29](#)
- Medium: Predictable pseudorandom number generator in [src/main/java/com/gitlab/security_products/tests/App.java:41](#)
- Medium: Predictable pseudorandom number generator in [src/main/java/com/gitlab/security_products/tests/App.java:47](#)

[Show complete code vulnerabilities report](#)

[Merge](#) Remove source branch Squash commits [Modify commit message](#)

4. SAST – Analyse statique

Limites du SAST

Bien qu'ayant un accès total au code, le SAST a plusieurs limitations :

- Les outils sont difficiles à intégrer / configurer. Souvent besoin d'un expert sécurité pour (au moins) **définir le bon seuil d'alerte** suivant la maturité sécurité de l'équipe de développement. Par défaut les outils ressortent tous les problèmes, cela mène à un **torrent d'infos qui sera forcément ignoré**.

→ Au début, ne lister que les vuln les plus graves. Faire plusieurs tests avec les Dev pour s'assurer que la longueur des premiers retours est limitée.

Puis relever, progressivement le niveau avec l'affichage de vuln moins graves.

4. SAST – Analyse statique

Limites du SAST

- Les résultats ne sont souvent pas directement compréhensibles par les Dev (qui ne savent pas, toujours, ce qu'est une XSS, SQLI, RCE ...).
- ➔ Reconfigurer les messages et surtout : faire pointer les vuln identifiées vers une base de connaissances qui explique comment concrètement corriger.

Si on ne peut pas facilement expliquer la correction ➔ il faut retirer l'affichage de cette vuln et la traiter par un accompagnement des Dev avec un expert sécu.

4. SAST – Analyse statique

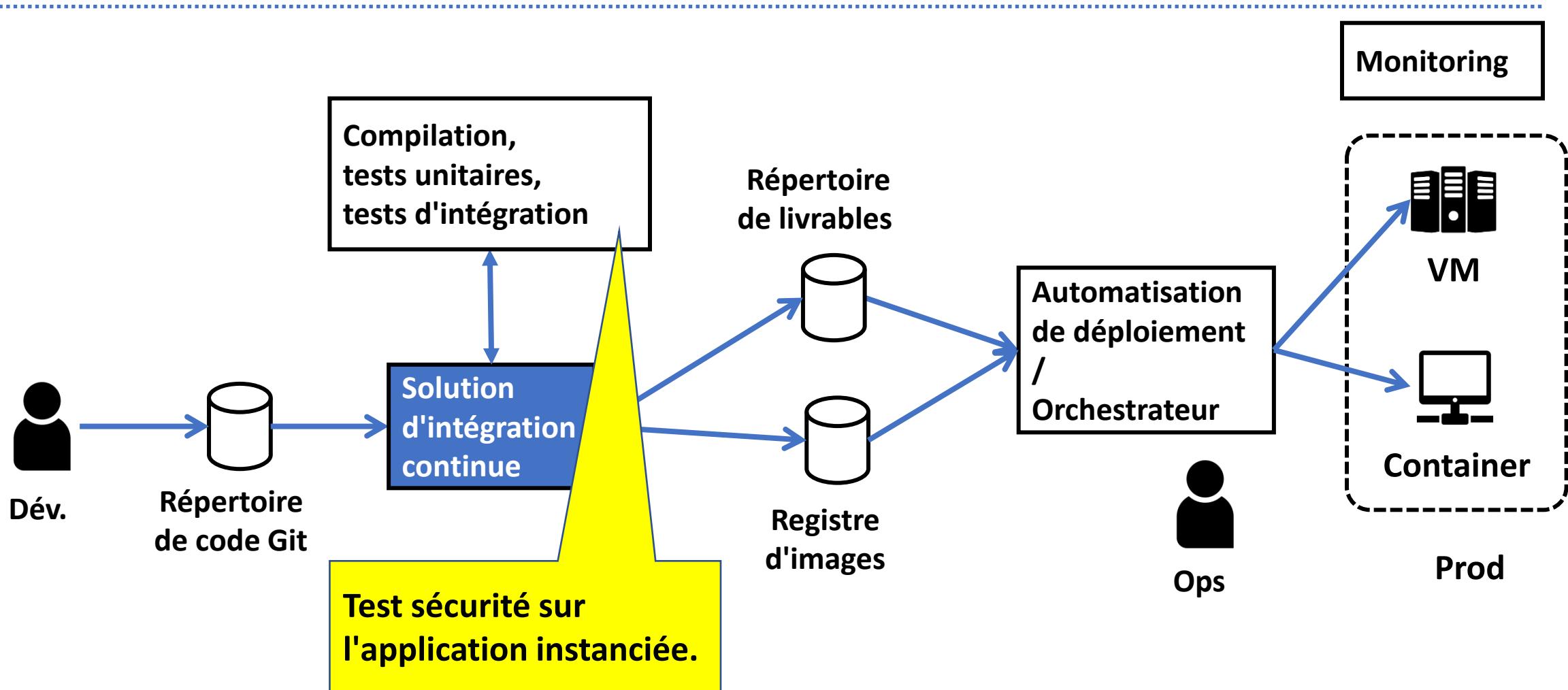
Limites du SAST

- Le SAST (et l'analyse automatisée) ne recherche que ce pour quoi on l'a programmé.
→ Il faut connaître les limites de l'outils et néanmoins savoir déjà identifier l'apport pragmatique qu'apporte un outil imparfait.
- Des vulnérabilités des serveurs et de l'infra ne seront pas identifiables avec le SAST (config TLS, exposition réseau, flags de cookies, header HTTP, ...)
→ Besoin de compléter l'analyse statique par d'autres approches.

5. DAST - Analyse dynamique

5. DAST - Analyse dynamique

Tests de sécurité



5. DAST - Analyse dynamique

Intro DAST

Dynamic Application Security Testing

Les tests automatisés sont exécuté sur une application instanciée (lancée).
C'est une approche « boîte noire ».

5. DAST - Analyse dynamique

Type de DAST

Types de tests :

- Injections (XSS, SQLI, XXE, ...)
- Scanner (ports ouverts, flags, entêtes, ...)
- Tests d'accès (un rôle de tel type a-t-il un accès légitime à telle ressource ?)

Cela se rapproche de ce que ferait un pentester en premiers tests basiques.

5. DAST - Analyse dynamique

Quelques outils de DAST

Outil de « référence » du DAST web / API, maintenu par l'OWASP : **ZAP Proxy**

ZAP Proxy est initialement connu pour son utilisation avec GUI.

Il existe une version CLI qui s'intègre dans une chaîne d'intégration continue.

5. DAST - Analyse dynamique

Quelques outils de DAST

Exemple de ZAP intégré dans Gitlab Enterprise.

Request to merge [add-dast](#)  into [master](#)

 Pipeline #58 passed for [7a941f11](#). 

 7 DAST alerts detected by analyzing the review app [Collapse](#)

- Low (Medium): Absence of Anti-CSRF Tokens
- Low (Medium): X-Content-Type-Options Header Missing
- Low (Medium): Password Autocomplete in Browser
- Low (Medium): Private IP Disclosure
- Informational (Medium): Information Disclosure - Suspicious Comments
- Medium (Medium): Application Error Disclosure
- Medium (Low): HTTP Parameter Override

 [Merge](#) Remove source branch Squash commits  [Modify commit message](#)

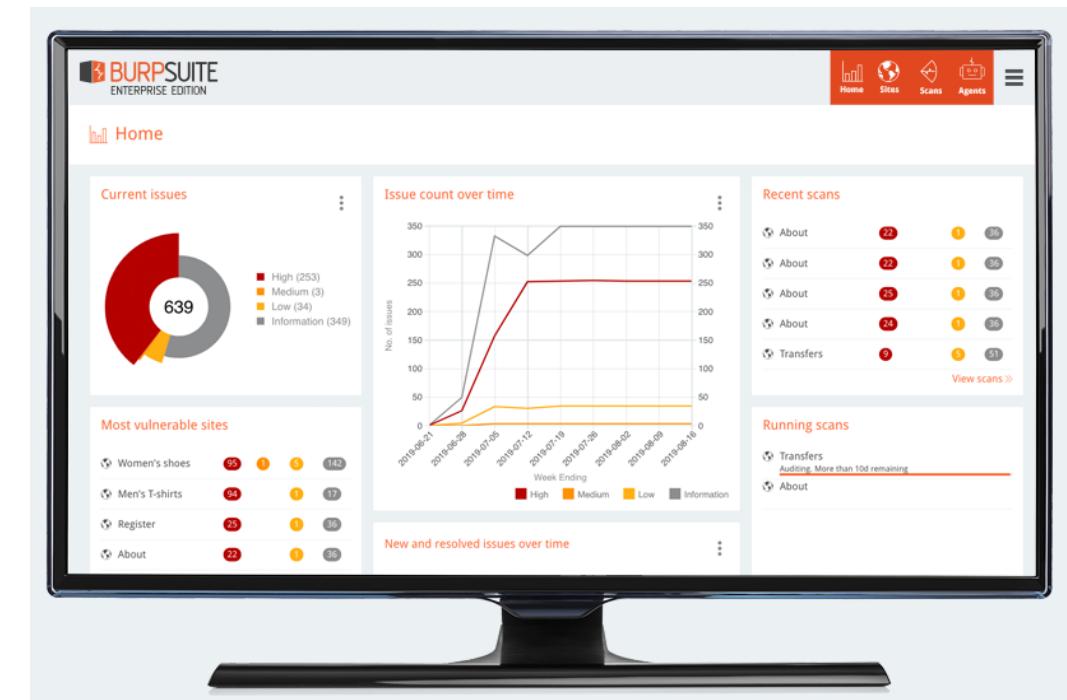
You can merge this merge request manually using the [command line](#)

5. DAST - Analyse dynamique

Quelques outils de DAST

Burp Suite Enterprise

Produit nouveau, peu de retours pour l'instant comparé à Burp Suite qui est une référence du pentest web.



5. DAST - Analyse dynamique

Les limites du DAST

- **Beaucoup de configuration initiale**
 - Endpoints HTTPS, swagger, user / pwd, rôles
 - Et comme en SAST : (au début) n'afficher que les vuln les plus graves et que l'on pourra concrètement corriger.
- Quand l'API ou les features changent ou sont ajoutées (ce qui est voulu dans un projet qui va bien), **il faut ajuster la config du DAST en continu.**

5. DAST - Analyse dynamique

SAST, DAST lequel faire en premier ?

Définitivement le SAST.

Cela demande déjà beaucoup d'itérations et configurations par des experts.

Le DAST s'envisage quand on commence à être mature sur le SAST.

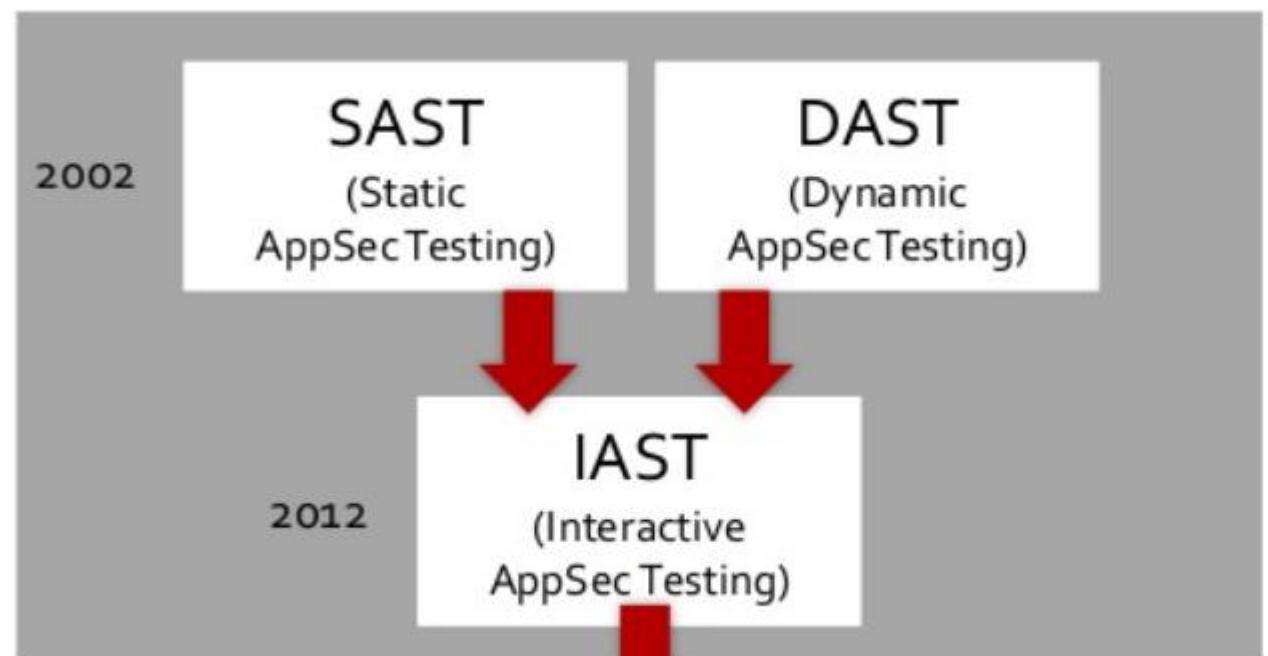
6. IAST

6. IAST

Interactive Application Security Testing

Combinaison du SAST et DAST pour améliorer la détection et réduire le taux de faux positifs.

Pour l'instant, il n'existe aucun produit open source pour le IAST.



<https://www.slideshare.net/planetlevel/continuous-application-security-at-scale-with-iast-and-rasp-transforming-devops-into-devsecops-64283374>

7. RASP

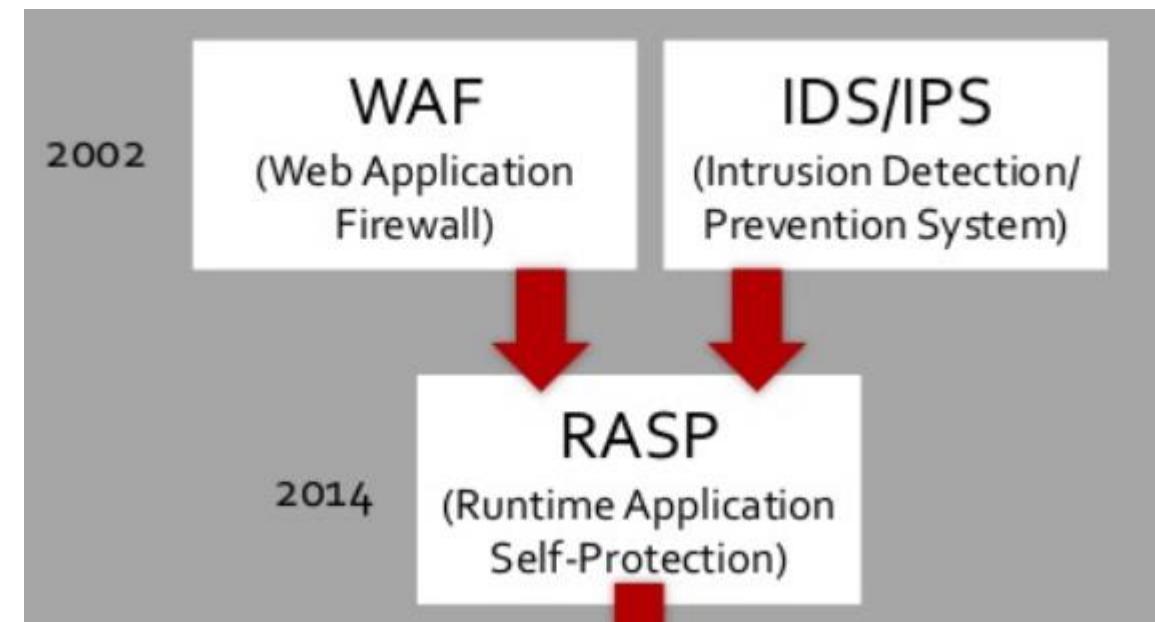
7. RASP

Runtime Application Self-Protection

Le RASP va détecter et bloquer les attaques lors de l'arrivée des requêtes et tentatives d'exécution de code.

Tous les langages ne sont pas encore supportés.

Pour l'instant, il n'existe aucun produit open source pour le RASP



8. Infrastructure as Code

8. Infrastructure as Code

Intro Infra as Code

Principe : gérer des ressources IT (config OS, VM, firewall, stockage, etc.) par l'intermédiaire de fichiers déclaratifs de descriptions.

Autrement dit : j'écris la cible à atteindre, puis la solution d'infra as code se débrouille pour l'atteindre.

8. Infrastructure as Code

Infra as Code VS. code impératif

"Oui, mais on a déjà {bash, Python, ...} pour faire cela".

- Alors, pas exactement.

En langage impératif, on indique quelle action réaliser.

Par ex. installer nginx : `apt install nginx`

A la différence du langage déclaratif, où on indique la cible à atteindre :

`package: name=nginx state=present`

8. Infrastructure as Code

Infra as Code VS. code impératif

L'intérêt du déclaratif par rapport à l'impératif :

1. D'abord, l'état actuel de la cible est testé.
2. Suivant le mode (juste tester ou appliquer), l'éventuel écart entre l'état déclaré et l'état actuel est corrigé.
(Autrement dit, le package est installé si et seulement s'il n'était pas déjà présent.)
3. Enfin, un rapport sur l'état initial, l'état final, l'éventuelle modification et/ou l'échec d'une modification est remonté par la solution d'Infra as code.

8. Infrastructure as Code

Plusieurs catégories d'Infra as code

1. Gestionnaire de configuration, provisioning de configuration.

Autrement dit du "config as code" utilisé pour configurer l'OS, les services (web, bdd, interpréteur) voire déployer du code.

→ Agit à l'intérieur de la VM.

Exemples de solutions : Ansible, Puppet, Chef.

8. Infrastructure as Code

Plusieurs catégories d'Infra as code

2. Gestionnaire d'infrastructure virtuelle.

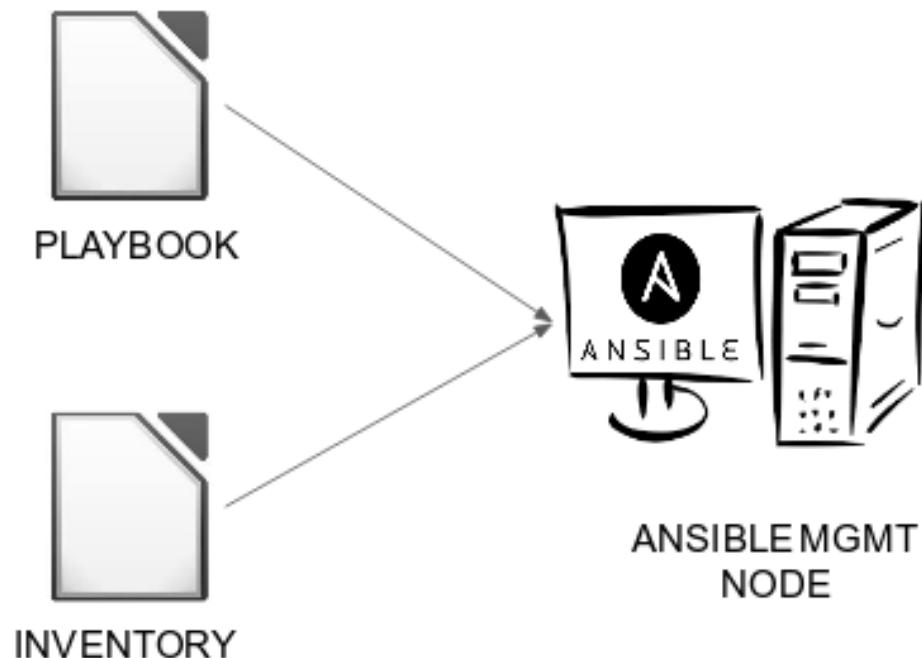
Instancie les VM, les firewalls (Security Group), la topologie réseau virtuelle, etc.

→ Agit donc à l'extérieur de la VM et bien au-delà.

Exemples de solutions : Terraform, CloudFormation (AWS).

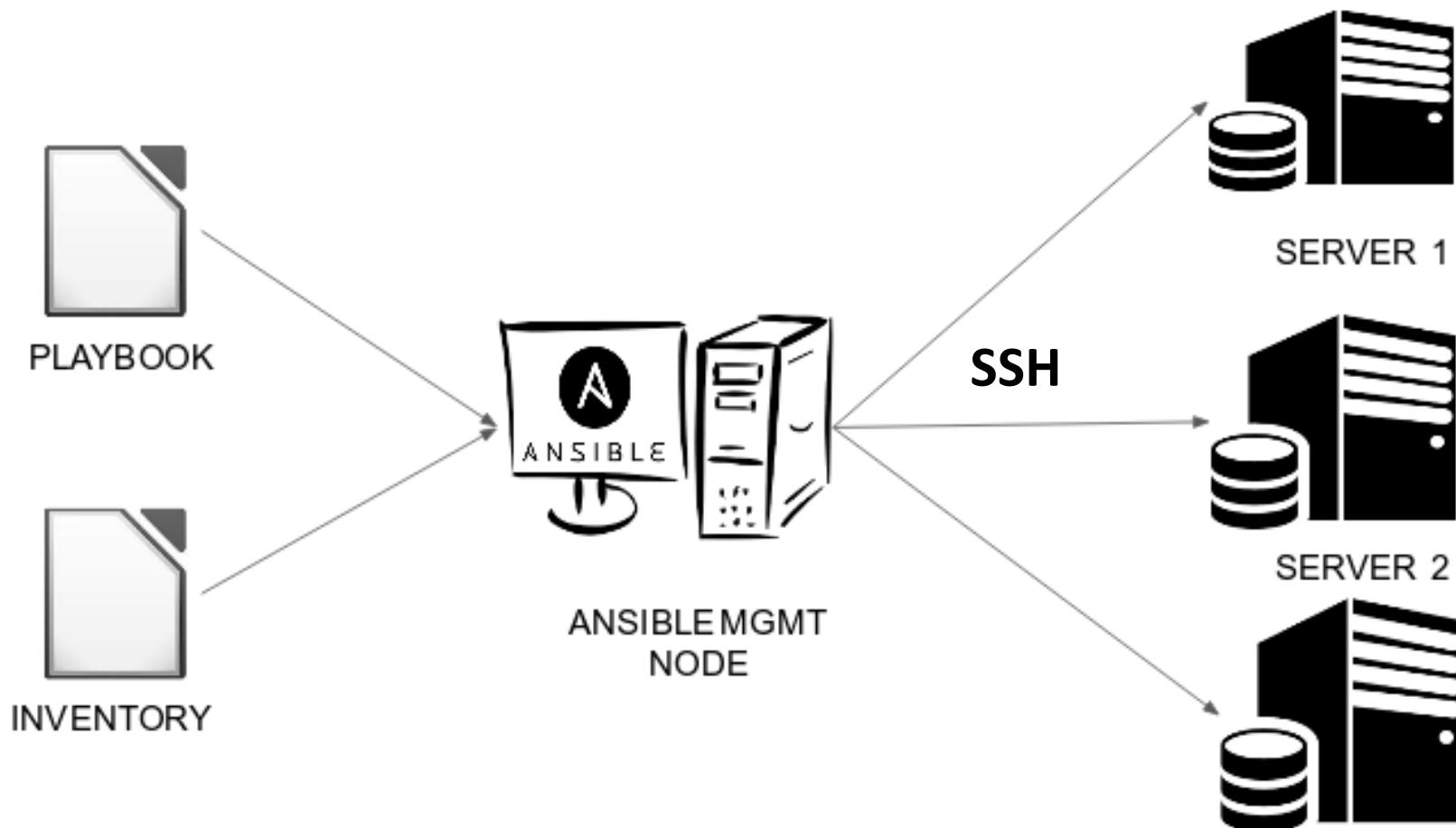
8. Infrastructure as Code

Ansible



8. Infrastructure as Code

Ansible



8. Infrastructure as Code

Avantage de l'Infra as Code

Gain pour le fonctionnel :

- Automatisation des déploiements → rapidité, productivité, scaling
- Application exacte de la cible déclarée (pas "d'erreur de frappe")
- Meilleure capitalisation de l'expérience pour améliorer les futurs playbooks
- Rollback (retour arrière) plus facile et rapide :
- Capacité à tester à l'identique sur une infra de test avant de pousser en prod.

Gain pour la sécurité :

- Déjà tous les points ci-dessous.

8. Infrastructure as Code

Avantage de l'Infra as Code

Gains additionnels pour la sécurité :

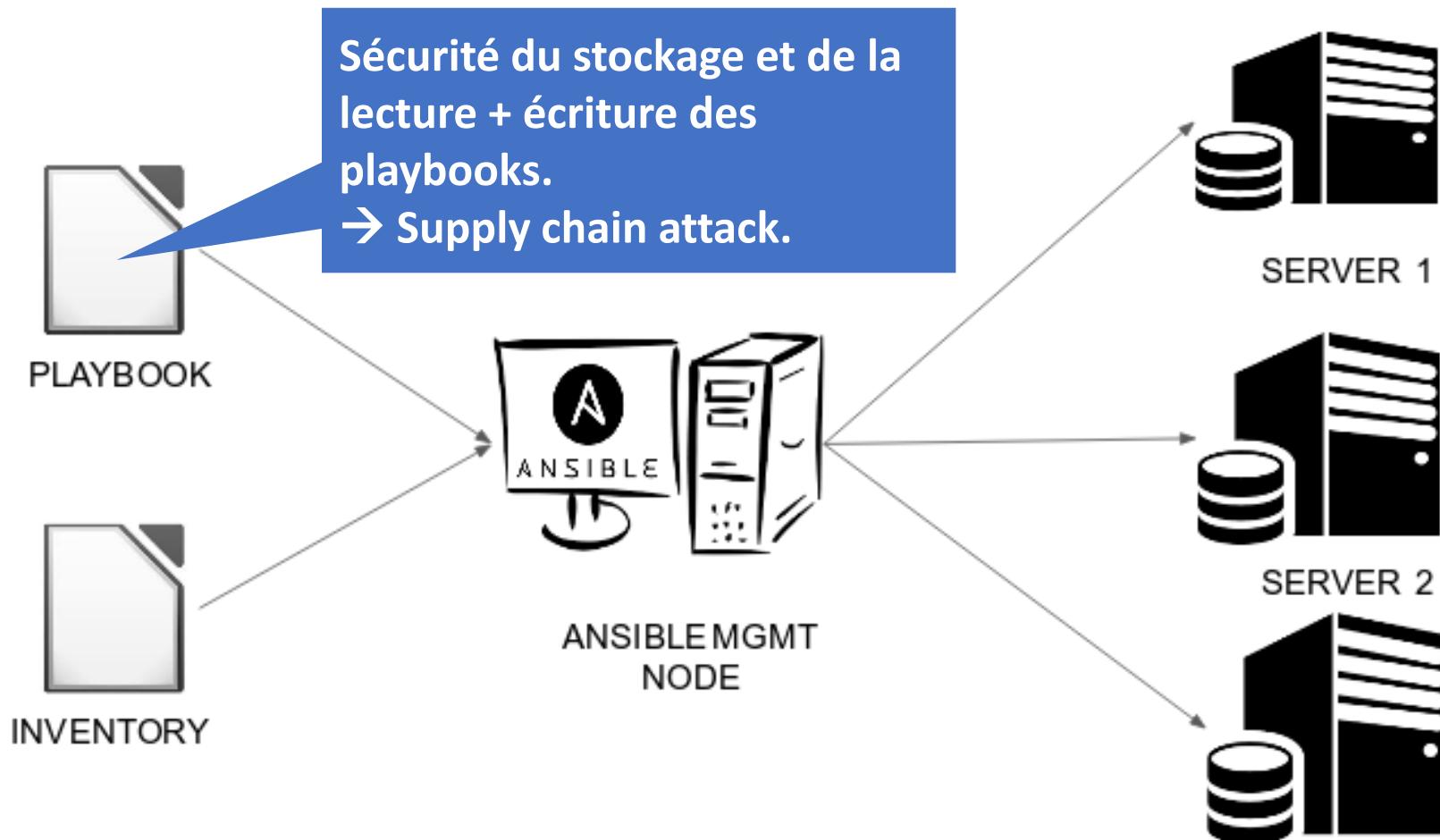
1. Capacité à automatiser les mises à jour logicielles.

2. Capacité à automatiser le renforcement de configuration :
Plutôt que d'écrire dans Word comment renforcer une VM, écrivons des playbooks de renforcement automatisé.

3. Possibilité pour la sécu d'aller faire une revue, voire proposer des modifications, dans les playbooks d'automatisation.

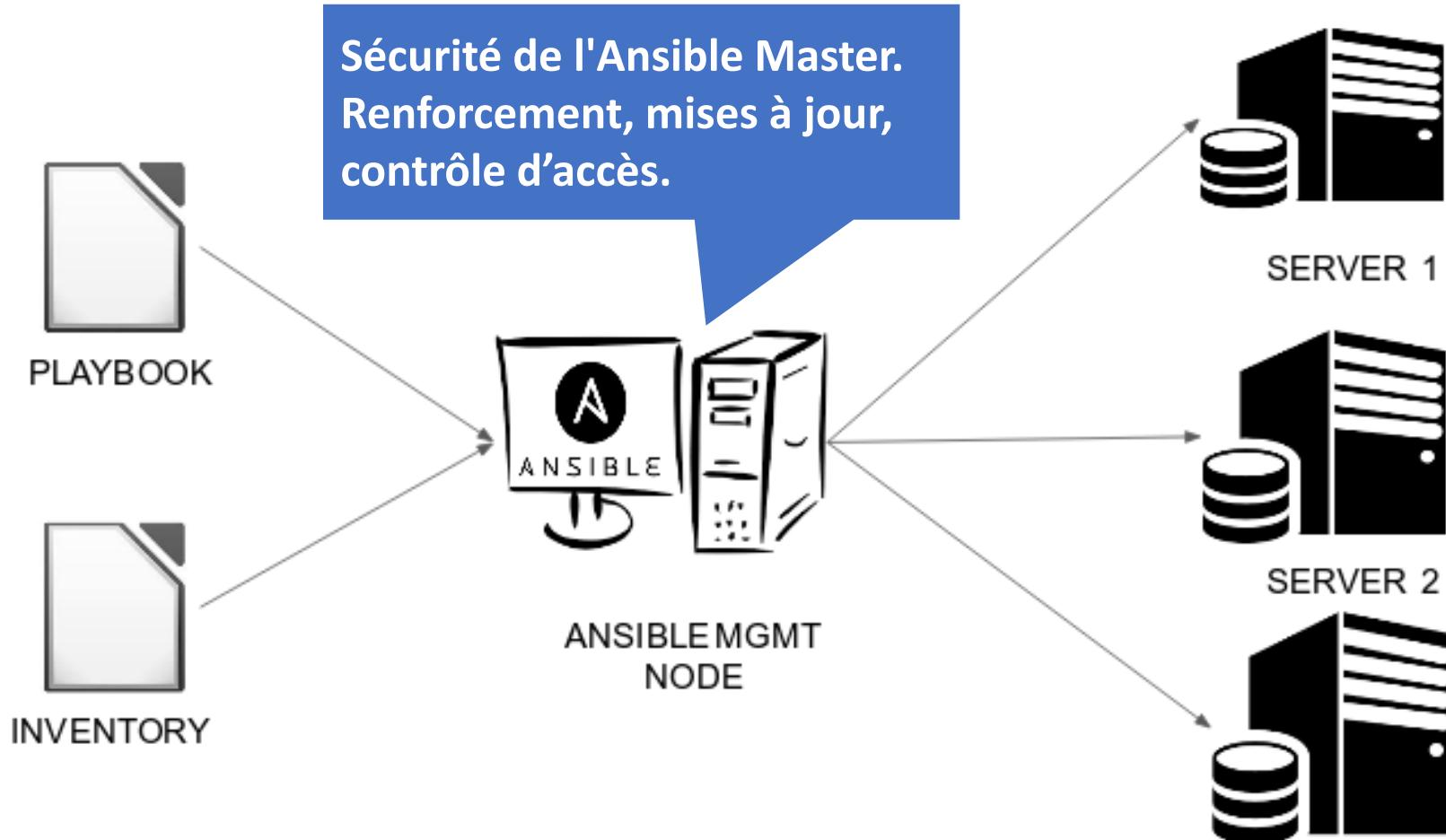
8. Infrastructure as Code

Risques et vigilances par vis-à-vis de l'infra as code



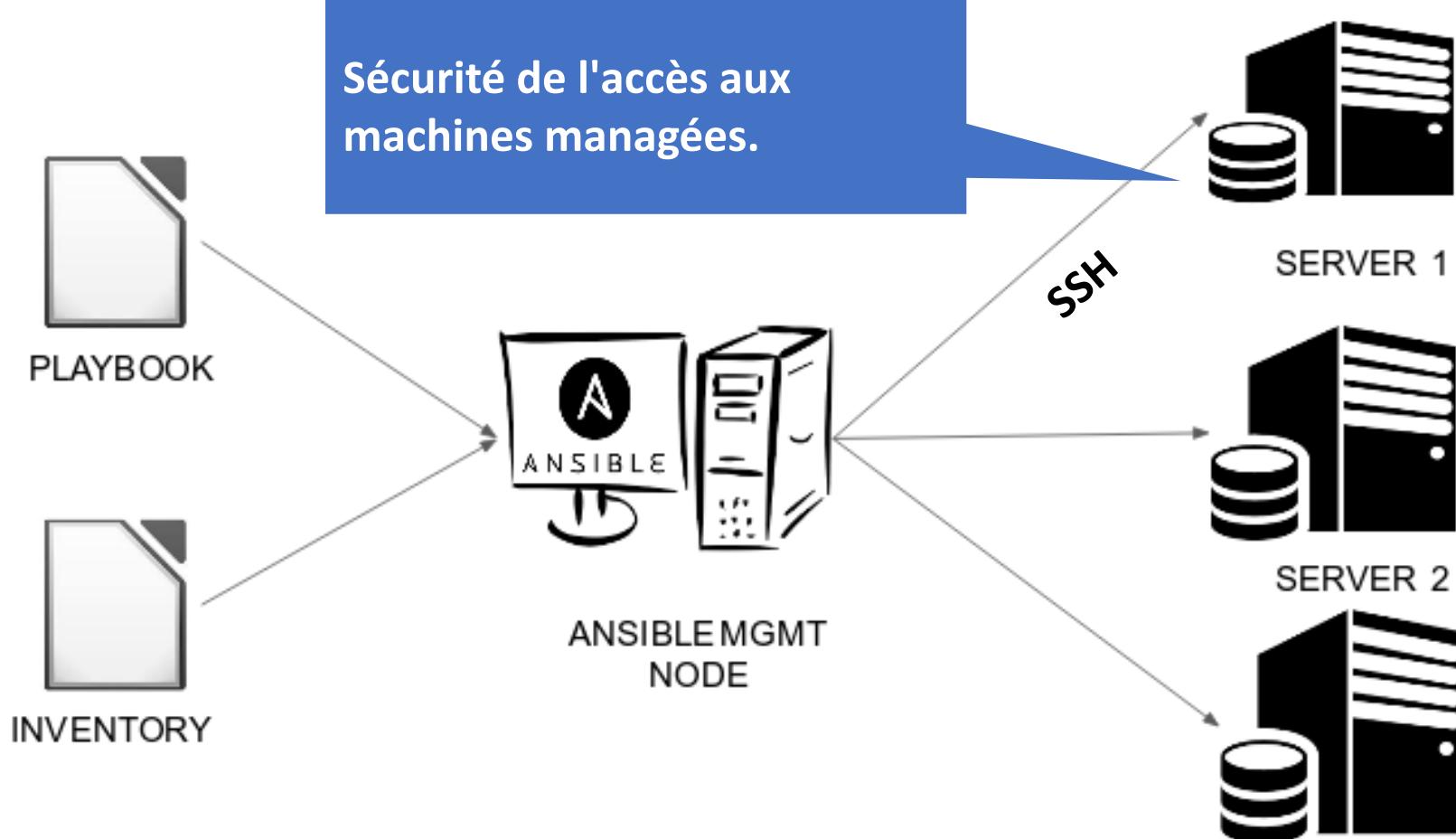
8. Infrastructure as Code

Risques et vigilances par vis-à-vis de l'infra as code



8. Infrastructure as Code

Risques et vigilances par vis-à-vis de l'infra as code



8. Infrastructure as Code

Ansible : rentrons dans le détail

```
[webservers]  
10.0.0.20
```

```
[databases]  
10.0.0.30
```

/etc/ansible/hosts

Utilisation de **variables**

Définition de **tâches** à accomplir

Les **handlers** permettent d'appeler certaines tâches

8. Infrastructure as Code

Ansible : rentrons dans le détail

```
---  
- hosts: webservers  
  vars:  
    http_port: 80  
    max_clients: 200  
  remote_user: root  
  tasks:  
    - name: ensure apache is at the latest version  
      yum:  
        name: httpd  
        state: latest  
    - name: write the apache config file  
      template:  
        src: /srv/httpd.j2  
        dest: /etc/httpd.conf  
      notify:  
        - restart apache  
    - name: ensure apache is running  
      service:  
        name: httpd  
        state: started  
  handlers:  
    - name: restart apache  
      service:  
        name: httpd  
        state: restarted
```

verify-apache.yml

Utilisation de **variables**

Définition de **tâches** à accomplir

Les **handlers** permettent d'appeler certaines tâches

8. Infrastructure as Code

Ansible : rentrons dans le détail

```
---  
- hosts: webservers  
  remote_user: root  
  
  tasks:  
    - name: ensure apache is at the latest version  
      yum:  
        name: httpd  
        state: latest  
    - name: write the apache config file  
      template:  
        src: /srv/httpd.j2  
        dest: /etc/httpd.conf  
  
- hosts: databases  
  remote_user: root  
  
  tasks:  
    - name: ensure postgresql is at the latest version  
      yum:  
        name: postgresql  
        state: latest  
    - name: ensure that postgresql is started  
      service:  
        name: postgresql  
        state: started
```

verify-apache.yml

Master SeCReTS - Mars 2020

On peut intégrer des tâches pour des **nœuds** ayant des fonctions différentes

8. Infrastructure as Code

Ansible : rentrons dans le détail

Les tâches s'exécutent dans l'ordre dans lequel elles ont été écrites.

Si un nœud rencontre une erreur, cela stoppe l'exécution.

On peut lancer ansible en mode « audit » : rien n'est modifié, listing des écarts entre la conf décrite et l'existant.

- `ansible-playbook foo.yml --check`

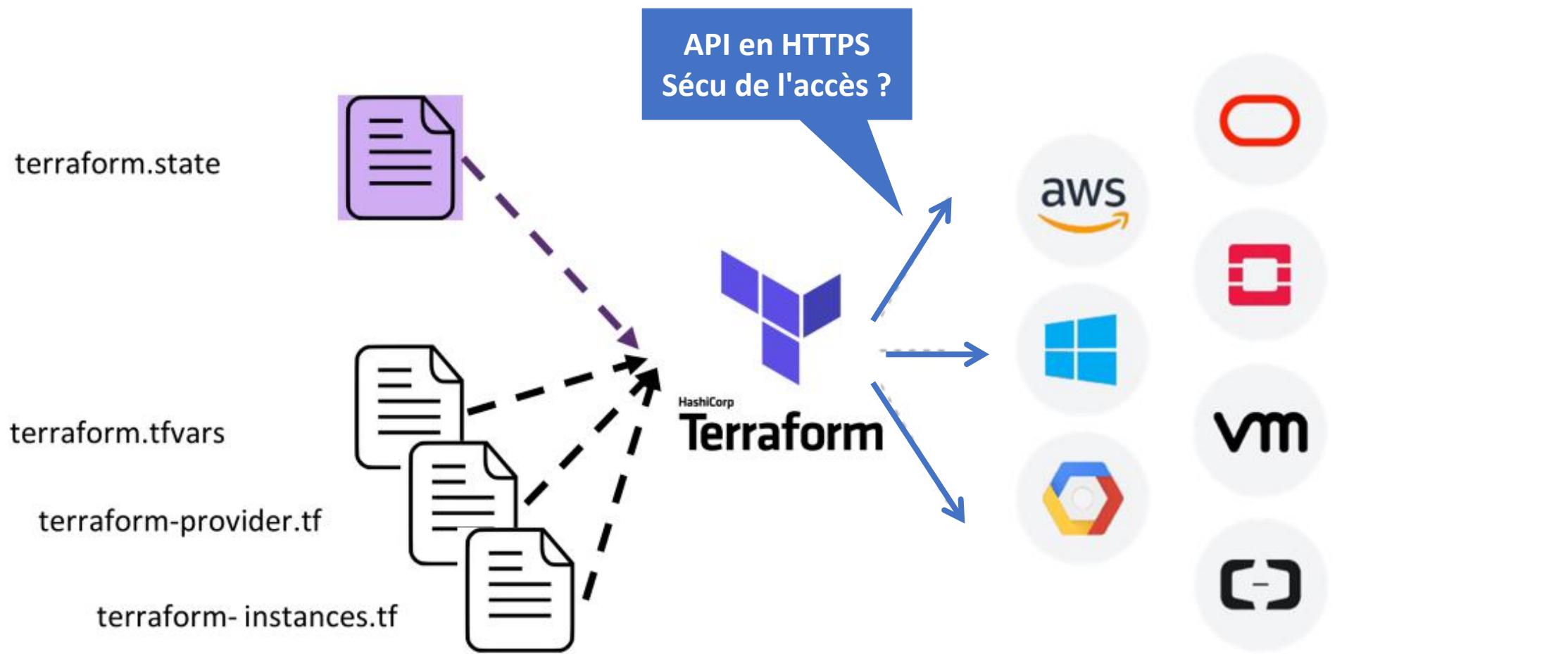
On peut lancer ansible en mode « modification » : les écarts entre la conf et l'existant sont corrigés.

- `ansible-playbook foo.yml`

De nombreux exemples : https://github.com/ansible/ansible-examples/blob/master/lamp_simple/README.md

8. Infrastructure as Code

Autre exemple : Terraform pour l'infra virtuelle



8. Infrastructure as Code

Infra as Code utilisé comme "Security as Code"

On peut doit (!) tirer partie de l'Infra as Code pour contrôler la sécurité.

On décrit la cible de sécurité à atteindre et l'outil vérifie les éventuels écarts.

Cela réalise un audit automatisé.

Des exemples de solutions : Inspec, Security Monkey (Netflix).

8. Infrastructure as Code

Exemple avec Inspec

```
describe file('/etc/sysconfig/init') do
  its('content') {should match 'umask 027'}
end
```

```
describe port(80) do
  it { should_not be_listening }
end

describe port(443) do
  it { should be_listening }
  its('protocols') {should include 'tcp'}
end
```

```
describe aws_security_group(group_name: linux_servers) do
  its('inbound_rules.first') { should include(from_port: '22',
ip_ranges: ['10.2.17.0/24']) }
end
```

9. Containers

9. Containers

Intro sur les containers

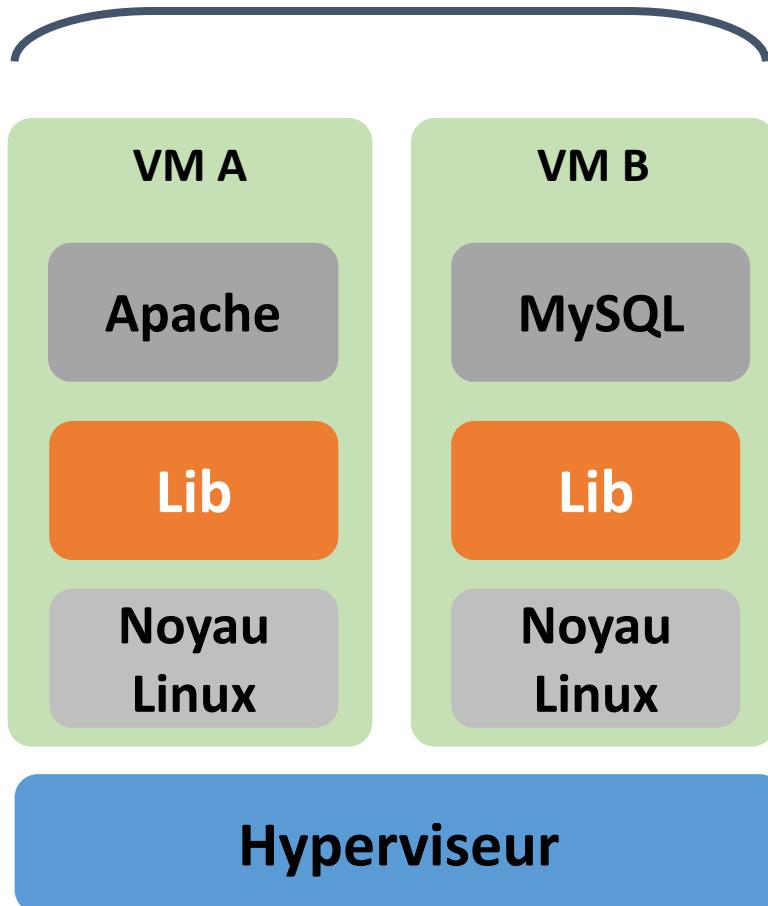
Avec l'émergence de Docker en 2014, les containers sont de plus en plus présents dans l'informatique "moderne".

Les containers n'ont pas été conçus pour la sécurité.

(Ce qui, de toute façon, est le cas pour la quasi-totalité de l'informatique)

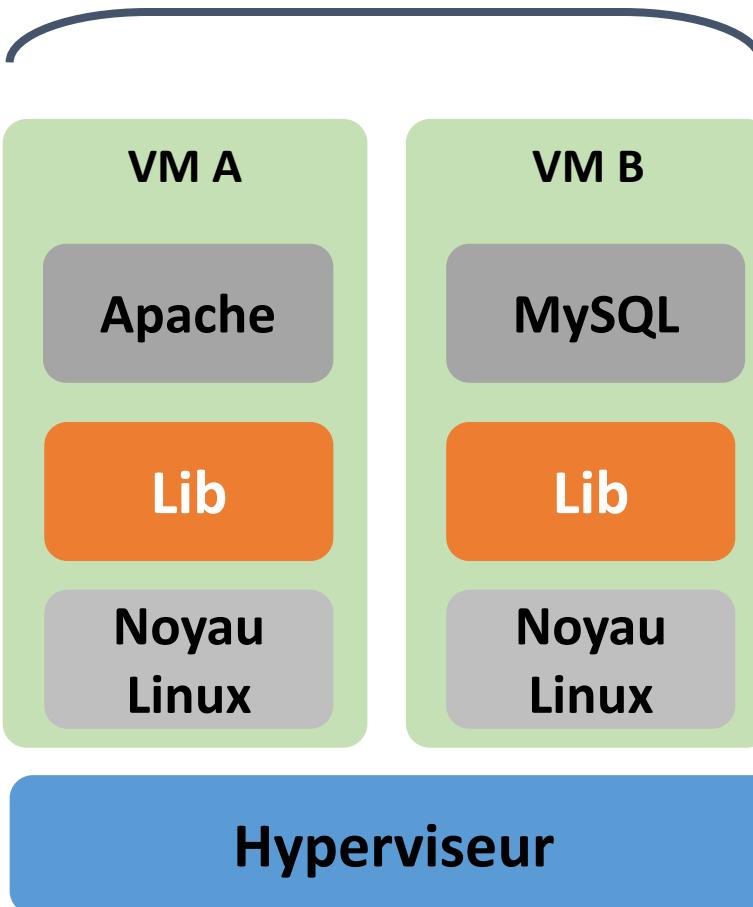
Ils font apparaître de **nouveaux risques**, mais en même temps de **nouvelles opportunités d'automatisation de la sécurité**.

VM : "*Full virtualization*"

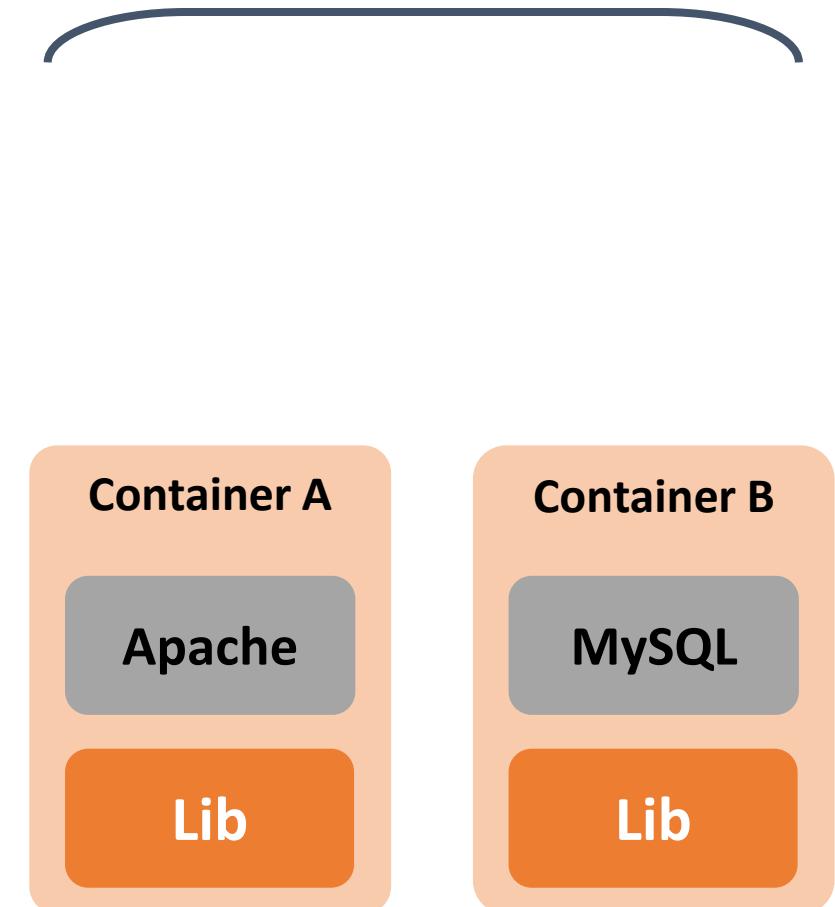


Noyau Linux

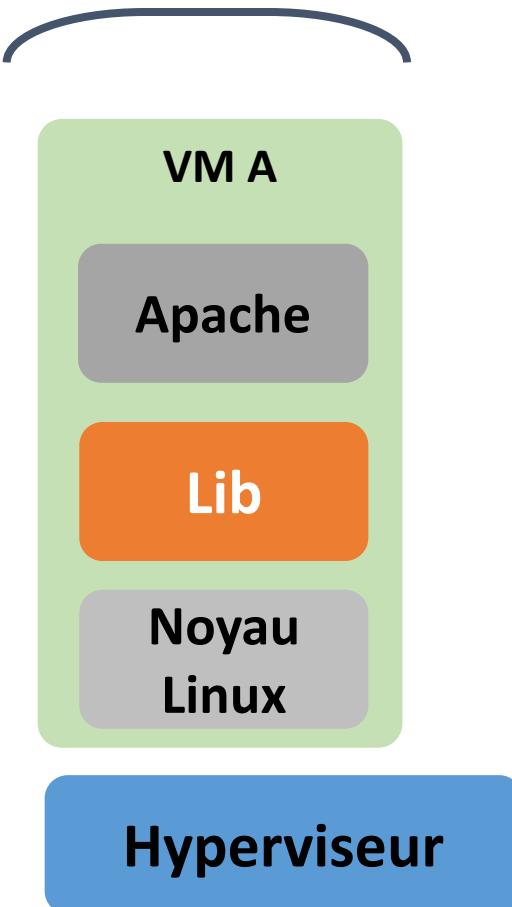
VM : "*Full virtualization*"



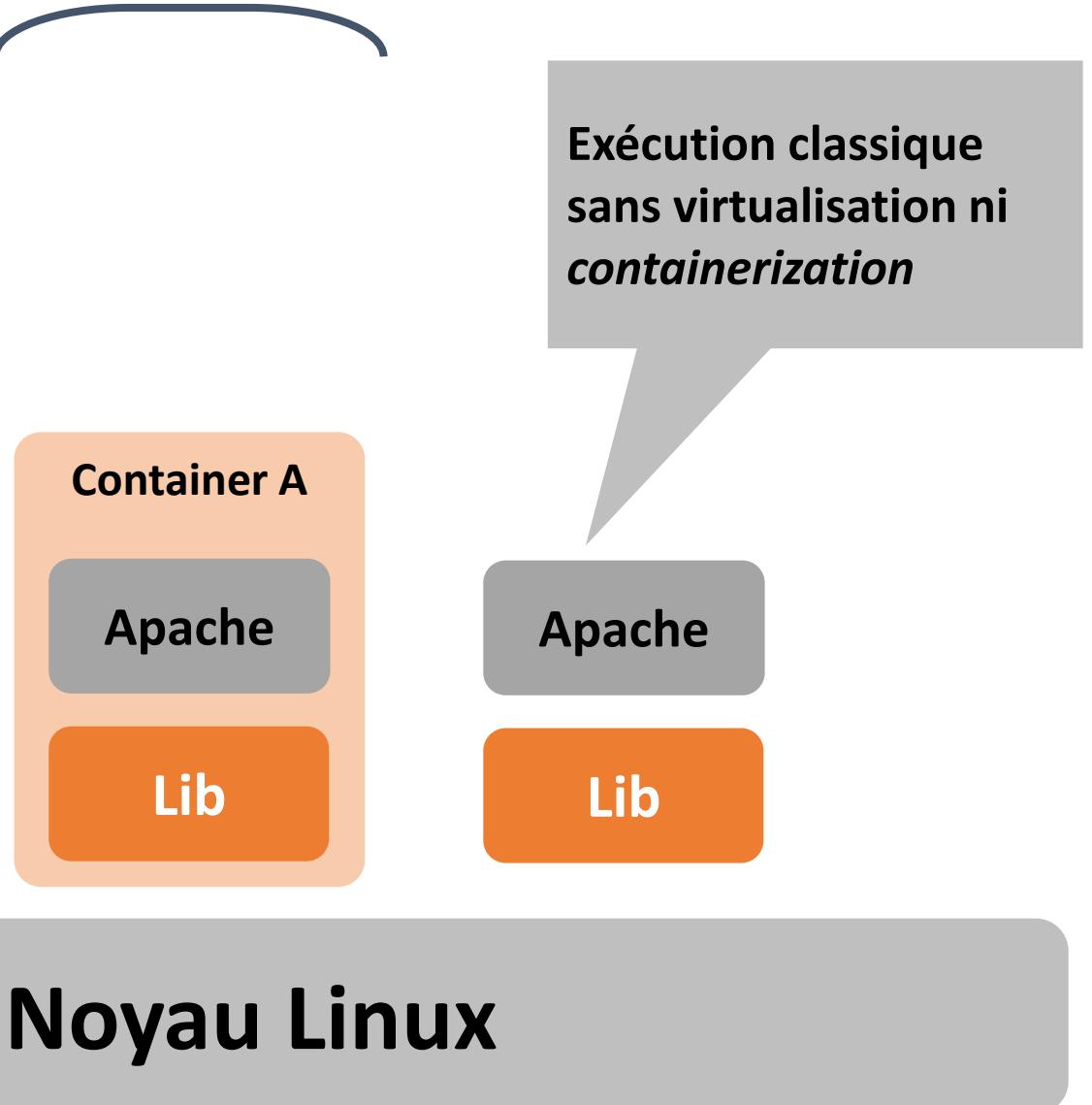
Containers : Virtualisation "lightweight" ou "system-level"



VM : "*Full virtualization*"



Containers : Virtualisation "*lightweight*" ou "*system-level*"



Exécution classique
sans virtualisation ni
containerization

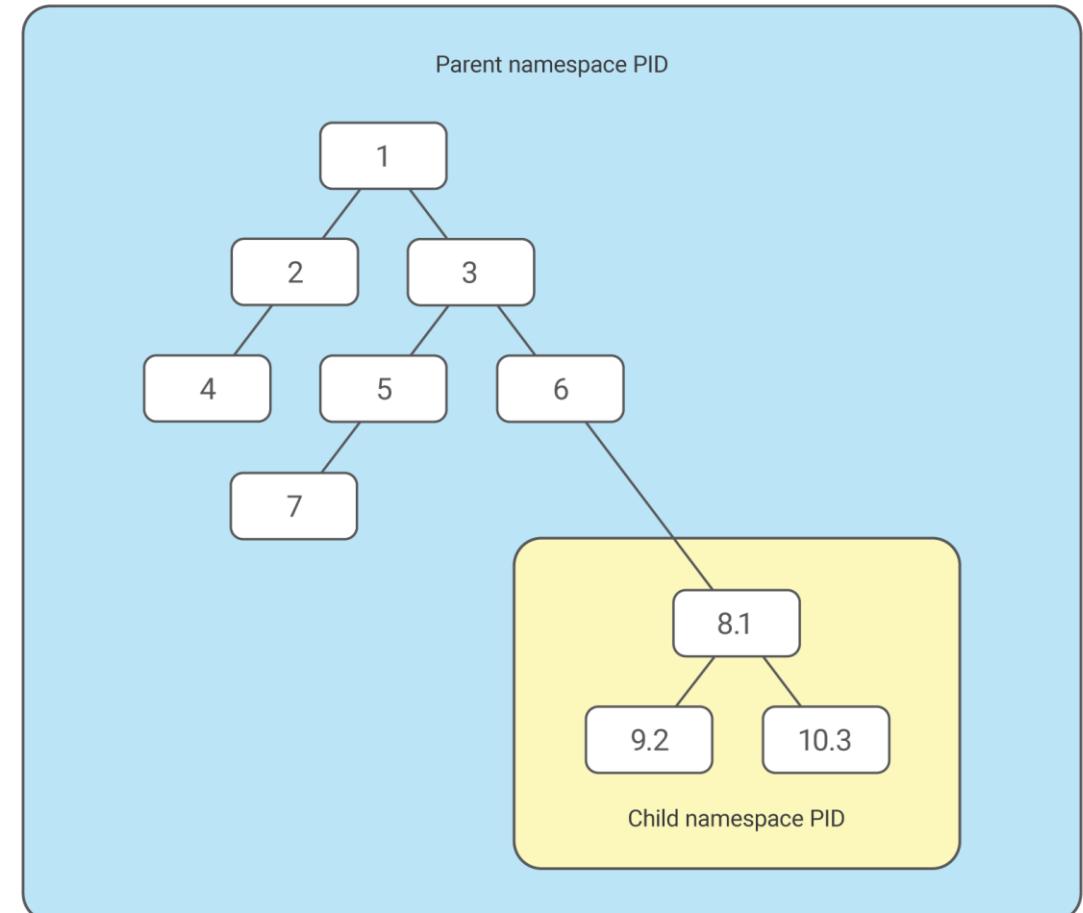
9. Containers

Techniquement

Les containers utilisent les ***namespaces*** du noyau Linux pour le cloisonnement des process.

En plus, ***cgroup*** est utilisé pour appliquer des limites à l'utilisation des ressources (CPU, RAM).

Aussi, il y a une suppression de certaines ***capabilities*** pour limiter les droits à l'intérieur des containers.



<https://blog.selectel.com/containerization-mechanisms-namespaces/>

what's a container?

a Linux container
is a group of processes



Container

we have our own
filesystem but
we're still just
regular processes!

Linux containers are
isolated from other processes
they can have their own:

- users
- network namespace
- filesystem
- process IDs
- memory / CPU limits

Kernel features that
isolate Linux containers

cgroups

namespaces

capabilities

seccomp-bpf

there are many ways
to run Linux containers

runc

Systemd-nspawn

LXC

Docker

your own homegrown
bash script

and containers can be
set up in different ways



Container 1

I have my own
filesystem!

I don't! I
have my system
calls restricted!



Container 2

extra confusion:

"container" sometimes means
"lightweight VM"

Fargate and kata Containers
are actually VMs and not
Linux containers (they don't
share a kernel with other
containers)

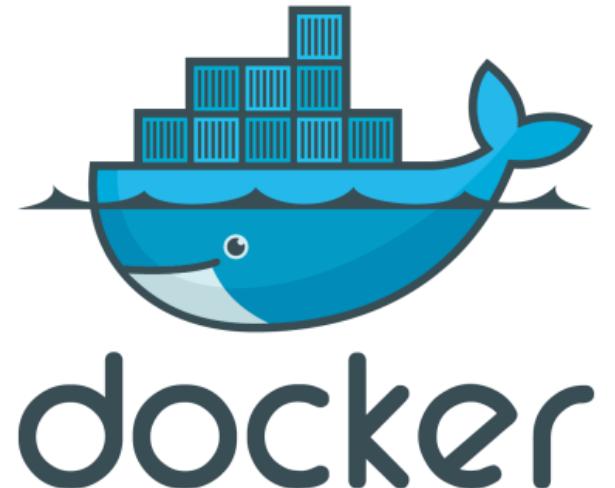
9. Containers

Docker

Docker est la solution de containerisation la plus utilisée.

Avant, il y a avait : LXC, V-Server...

Après, il y aura (peut-être) : CRI-O, Podman, Kata, gVisor...

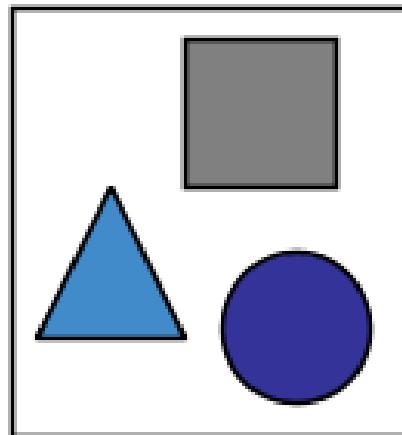


9. Containers

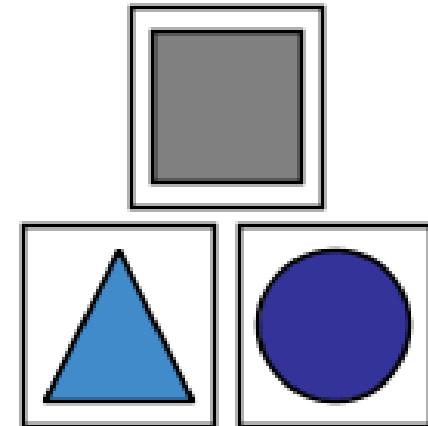
Pourquoi les containers ?

Principe de découplage

Monolith



Microservices



□ Process

9. Containers

Apport des containers

Les containers permettent de réduire l'adhérence entre services.

→ Facilite l'évolution / patching car le couplage entre service est moindre.

Accélérer le (re)déploiement et maintenance.

"Densifier" le déploiement → Davantage de services sur le même matériel.

Faciliter l'orchestration et le scaling.

9. Containers

Approche image

Récupération des images Docker (images de containers) sur un registre d'image.

```
docker pull centos
```



OFFICIAL REPOSITORY

centos

Last pushed: 4 days ago

OFFICIAL REPOSITORY

mysql

Last pushed: 2 days ago

OFFICIAL REPOSITORY

debian

Last pushed: 15 days ago

9. Containers

Espace quasi mono-process

Exemple d'un container MySQL

```
root@10c64da891d1:/srv# ps -edf
UID        PID  PPID  C STIME TTY          TIME CMD
root            1      0  0 Feb10 ?          00:00:00 /bin/bash /usr/bin/mysqld_safe
mysql         355      1  0 Feb10 ?          10:32:22 /usr/sbin/mysqld --basedir=/usr --dat
root            415      0  0 Feb10 ?          00:00:00 /bin/bash
root            428      0  0 Feb10 ?          00:00:00 /bin/bash
root          4585     428  0 17:03 ?          00:00:00 ps -edf
root@10c64da891d1:/srv#
```

9. Containers

Dockerfile pour une construction déterministe

Dockerfile (tronqué) de Wordpress

```
FROM php:7.3-apache

RUN set -ex; \
curl -o wordpress.tar.gz -fSL "https://wordpress.org/..."

COPY docker-entrypoint.sh /usr/local/bin/
ENTRYPOINT ["docker-entrypoint.sh"]
```

Image de base

Téléchargement des sources (à chaque build)

Définit le fichier à exécuter au lancement du container

Copie (ajoute) un fichier dans le container

9. Containers

Quelques principes

Les containers n'exécutent qu'un seul service (web, base de données, etc.).

→ Ainsi, jamais de serveur SSH dans un container (ni aucun agent de monitoring).

Aucune modification (upgrade package ou code) n'est faite dans un container.

Une modification du container implique la reconstruction de l'image du container.

→ Aucune mise à jour sécurité à l'intérieur d'un container.

Il faut reconstruire l'image pour appliquer une M&J sécurité.

9. Containers

Les risques des containers ?

De nouveaux risques :

- L'image contient-elle des vulnérabilités ?
- L'image récupérée est-t-elle bienveillante ?
- En cas de compromission d'un container, est-ce que l'hôte arrivera à éviter la propagation dans les autres containers ?
- En cas de compromission d'un container, comment peut-on investiguer ? (prévoir les métriques & les logs en amont)
- Quelle est l'exposition réseau des containers ?

9. Containers

Maîtriser les risques

L'image contient-elle des vulnérabilités ?

Utiliser des outils de scan d'images.

Plusieurs opensource:

- Trivy
- Clair
- Anchore

ubuntu:xenial-20190515 (ubuntu 16.04)			
=====			
Total: 153 (UNKNOWN: 0, LOW: 11, MEDIUM: 102, HIGH: 40, CRITICAL: 0)			
LIBRARY	VULNERABILITY ID	SEVERITY	INSTALLED VERSION
bsdutils	CVE-2016-2779	HIGH	2.27.1-6ubuntu3.6
	CVE-2016-5011	MEDIUM	
coreutils	CVE-2016-2781	LOW	8.25-2ubuntu3~16.04
dpkg	CVE-2017-8283	HIGH	1.18.4ubuntu1.5
libapparmor1	CVE-2016-1585		2.10.95-0ubuntu2.10
libblkid1	CVE-2016-2779		2.27.1-6ubuntu3.6

9. Containers

Maîtriser les risques

L'image récupérée est-t-elle bienveillante ?

Peut-on faire confiance à tout ce que l'on télécharge sur Internet ?

- Il faut douter des images sur les registres publics (Dockerhub, Quay)
- Reconstruire ses propres images à partir des images officielles

Le niveau du dessus :

- Signer ses images et interdire l'exécution en prod d'images non-signées.

Tesla cloud resources are hacked to run cryptocurrency-mining malware

Crooks find poorly secured access credentials, use them to install stealth miner.

DAN GOODIN - 2/20/2018, 8:21 PM

9. Containers

Maîtriser les risques

En cas de compromission d'un container, est-ce que l'hôte arrivera à éviter la propagation dans les autres containers ? (1/2)

- Auditer la machine ou cluster Docker, Kubernetes, OpenShift... sous-jacente pour vérifier les mises à jour et la configuration renforcée.
- Prévoir un process de patching (sans interruption des services hébergés).

Comme le noyau est partagé entre les containers d'un même hôte, une vulnérabilité noyau peut entraîner la compromission des autres containers de l'hôte, voire du cluster entier.

Scripts de test pour hôtes : <https://www.cisecurity.org/benchmark/docker/>

9. Containers

Maîtriser les risques

En cas de compromission d'un container, est-ce que l'hôte arrivera à éviter la propagation dans les autres containers ? (2/2)

Pour éviter une « *container-escape* »

- Mettre à jour le **noyau** !
- **Ne jamais** exécuter le container en mode *privileged*.
- Configurer les ***capabilities*** à retirer au container.
- Bonus : Configurer **SELinux** ou **AppArmor** (restriction d'accès aux fichiers)
- Bonus : configurer seccomp-BPF (filtrage d'appels système)

9. Containers

Maîtriser les risques

Quelle est l'exposition réseau des containers ?

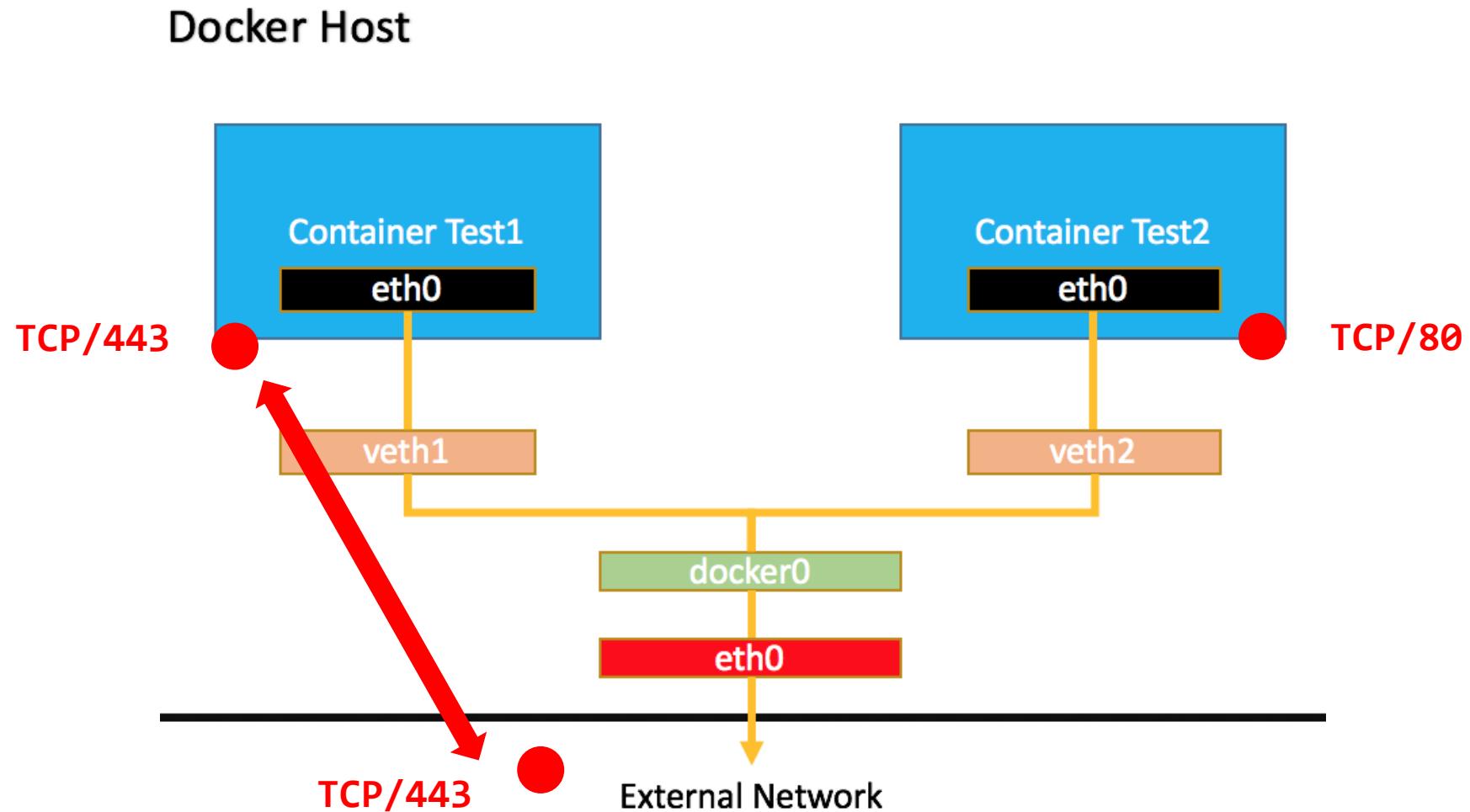
Un container s'expose de deux façons :

- Dans le Dockerfile : la directive `EXPOSE 80` va exposer le port TCP 80 aux autres containers de l'hôte. Mais pas d'exposition "externe" à l'hôte.
- Au niveau de l'orchestrateur : `--port 443:443` va mapper le port 443 de l'hôte sur le port 443 du container. Ainsi, le container va se retrouver exposé aux machines en réseau avec l'hôte (voire Internet ?).

- Ne pas exposer les services "internes" sur les ports de l'hôte.
- Et déjà, est-on confiant dans l'exposition interne d'un container si son voisin se fait compromettre ?

9. Containers

Schéma réseau



9. Containers

Service Mesh

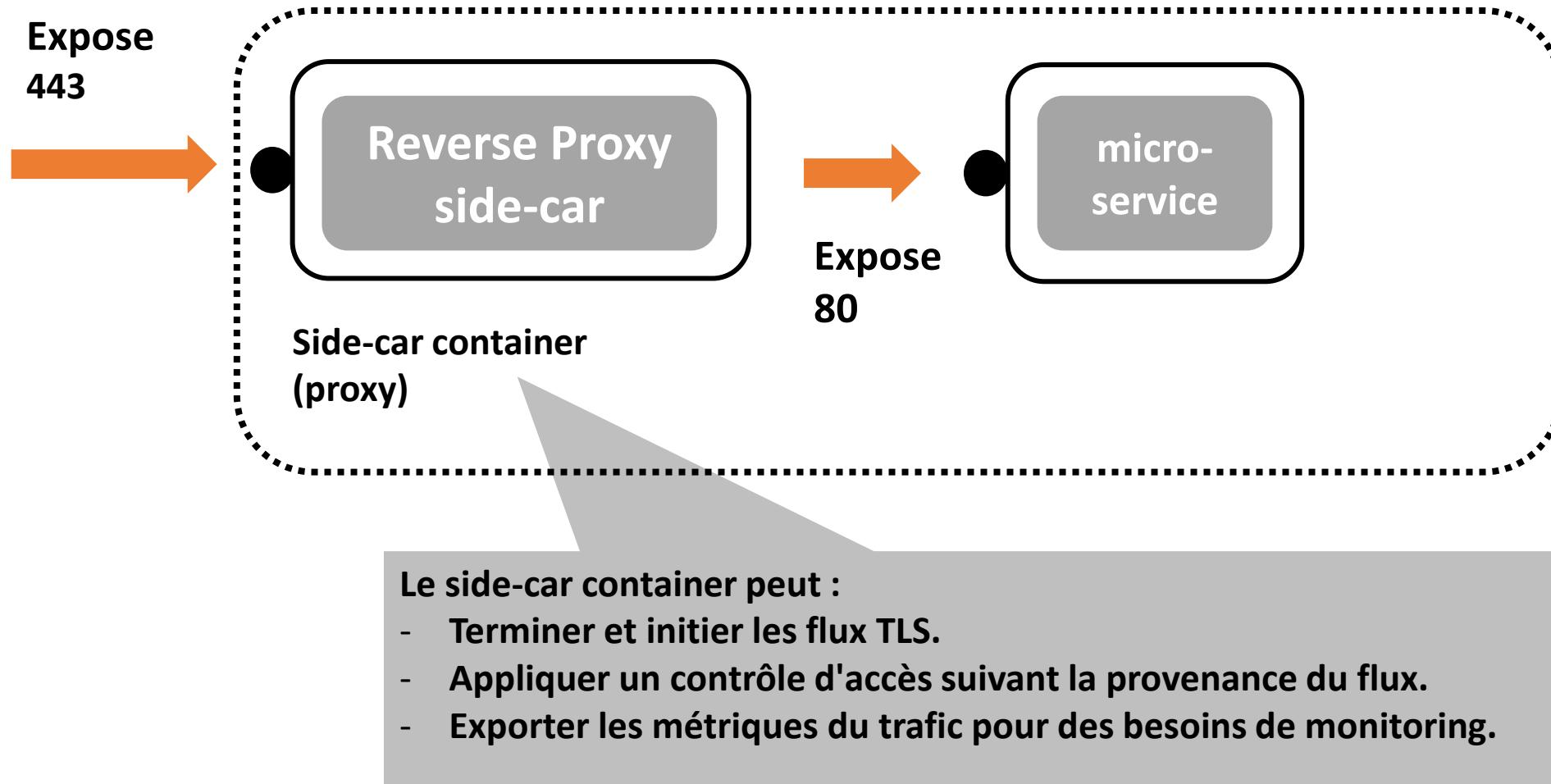
Le Service Mesh commence à émerger pour proposer de la sécurisation fine des communications entre containers :

- Notion de cloisonnement réseau au niveau applicatif entre containers.
- Chiffrement, par défaut, de tous les flux entre containers.

Exemple de solutions : Istio, Linkerd...

9. Containers

Notion de side-car containers pour le chiffrement "by default"



10. Gestion des secrets

10. Gestion des secrets

Intro

Lorsque que l'on automatise les déploiements, où stocker les mots de passes, clés, tokens, etc ?

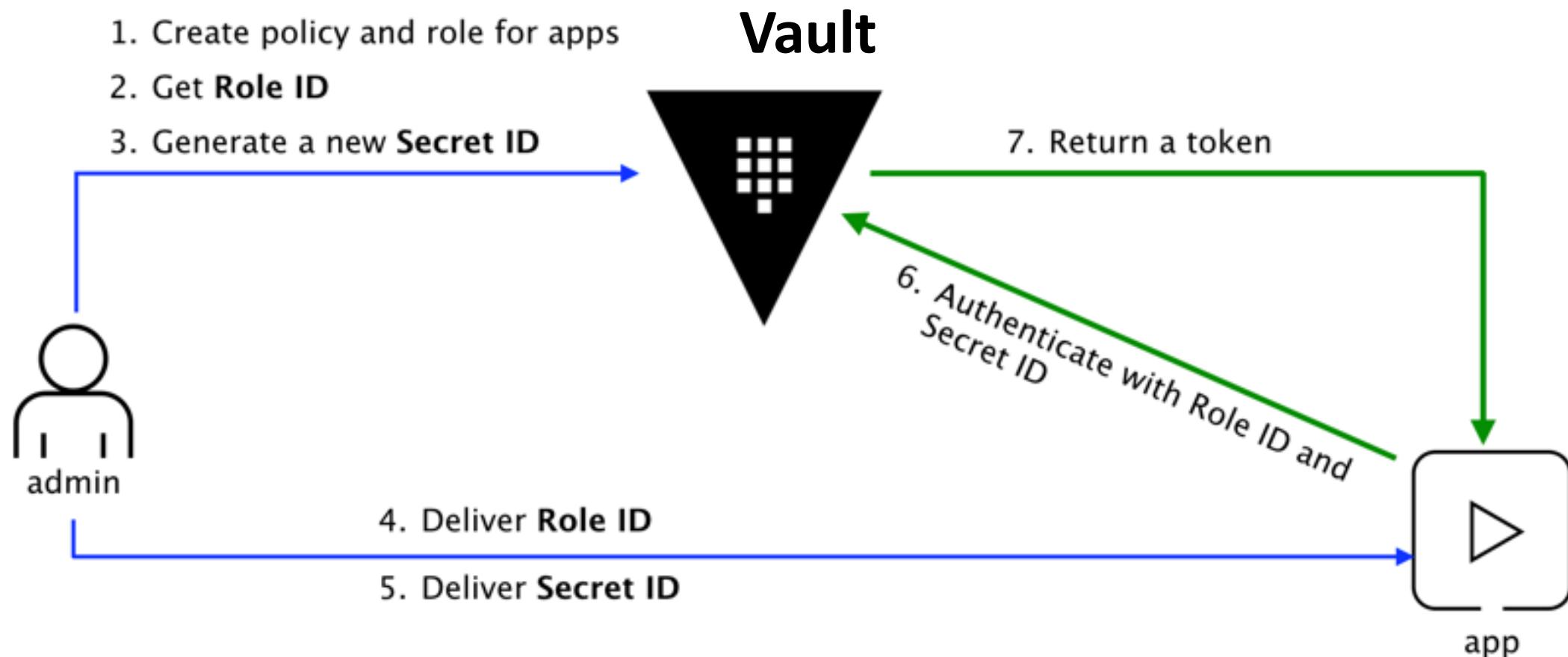
- Dans un playbook stocké dans Git → mauvaise idée !

À la place, un coffre-fort à secrets va :

- Générer les secrets (clés, tokens, mots de passe).
- Permettre aux services de venir récupérer leur secret.

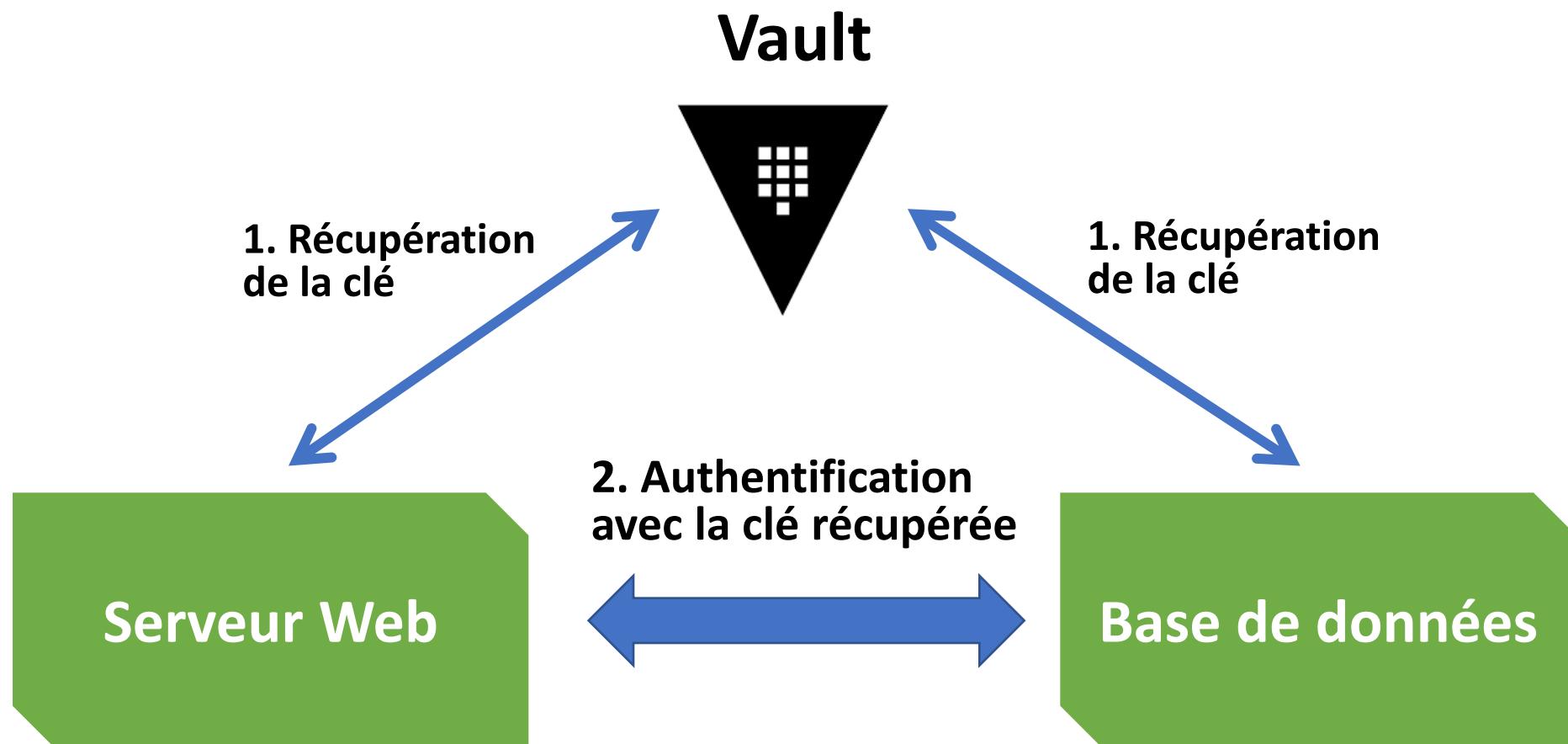
10. Gestion des secrets

Exemple de Vault Hashicorp



10. Gestion des secrets

Ainsi entre deux services



10. Gestion des secrets

D'autres cas d'usage pour la sécurité DevOps

Ce type de Vault permet également d'adresser les cas d'usages suivants :

- Génération de **PKI X.509** (certificats HTTPS).
 - → Générer les certificats HTTPS dynamiquement.
- Génération de **PKI certificats SSH** (qui ne sont pas compatibles X.509).
- Brique de **chiffrement as a service**.

11. Pour aller plus loin

11. Pour aller plus loin

Web

Série de 23 articles « Pushing Left, Like a Boss » de Tanya Janca (SheHacksPurple)

<https://dev.to/azure/pushing-left-like-a-boss-part-1-4d9i>

11. Pour aller plus loin

Comptes Twitter

<https://twitter.com/shehackspurple>

<https://twitter.com/lizrice>

<https://twitter.com/clintgibler>

<https://twitter.com/jvehent>

<https://twitter.com/manicode>

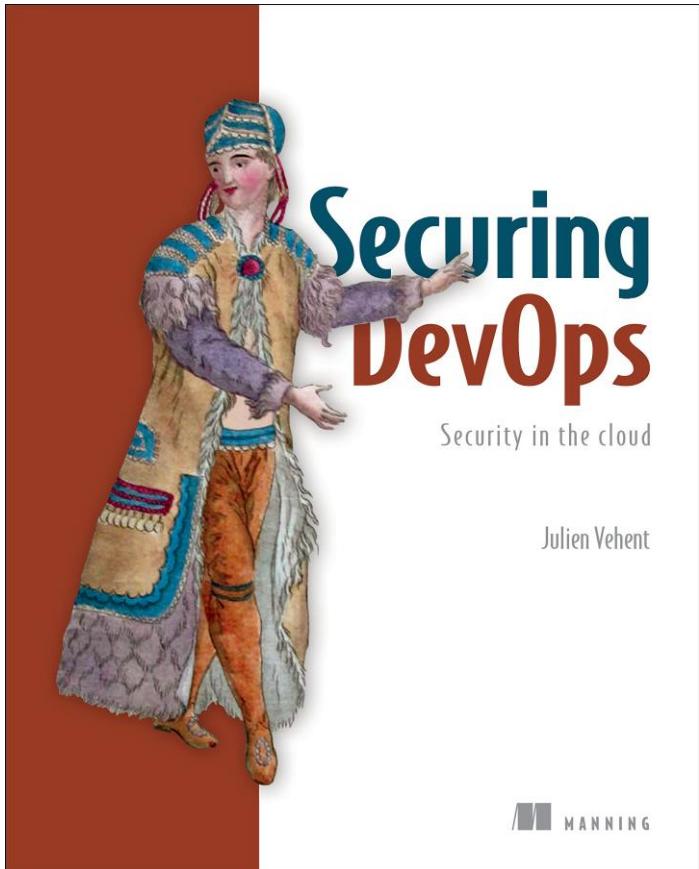
<https://twitter.com/TeriRadichel>

<https://twitter.com/dinodaizovi>

<https://twitter.com/b0rk>

11. Pour aller plus loin

Ouvrages



Laura Bell, Michael Brunton-Spall,
Rich Smith & Jim Bird