Tech Radar

#2

OCTOBRE 2024

theodo. Cloud

INFRASTRUCTURE & CLOUD

Définir, construire et améliorer votre infrastructure

Introduction

"La simplicité est la sophistication suprême."

LÉONARD DE VINCI

Naviguer dans le vaste et complexe univers des technologies Cloud peut vite paraître déroutant. Avec cette nouvelle édition de notre Tech Radar, notre mission est de vous accompagner dans ce voyage en vous donnant notre vision du paysage du Cloud. Ce document est le résultat de nos recherches approfondies, de réflexions collectives et de débats nourris, issus de notre expérience terrain bâtie grâce à la grande diversité de nos clients.

Ce Tech Radar ne prétend pas détenir une vérité absolue ni offrir une réponse unique à tous vos enjeux. Il s'agit plutôt d'une compilation des choix technologiques qui, selon nous, représentent le meilleur équilibre entre sophistication technique et simplicité fonctionnelle. Chaque technologie et chaque pattern sont évalués en fonction de leur pertinence et de leur efficacité, basées sur des expériences concrètes et des retours terrain.

Cependant, nos recommandations vont au-delà du choix des meilleures technologies pour accélérer la construction de vos architectures Cloud. Notre expérience d'infogérant nous permet d'évaluer la durabilité des choix technologiques dans le temps.

Nous vous invitons à tester ces technologies et à éprouver nos opinions, nous sommes convaincus que c'est par la pratique, l'expertise et les débats que naissent les solutions les plus pragmatiques et durables.

Bonne lecture !

SOMMAIRE

Le Tech Radar

Ce Tech Radar regroupe une soixantaine de technologies Cloud et DevOps éprouvées par les experts de Theodo Cloud durant plus de 5 ans de projets. Elles sont regroupées en 4 cadrans : Resilient, Operable, Secure et Empowering.

Ces 4 cadrans composent ROSE, notre framework permettant de mesurer la qualité des infrastructures que nous construisons et opérons au quotidien. ROSE signifie :

- **RÉSILIENTE :** l'infrastructure permet une haute disponibilité du service pour l'utilisateur final
- **OPÉRABLE :** l'infrastructure est facilement maintenable par les DevOps
- **SÉCURISÉE :** l'infrastructure présente une très faible surface d'attaque
- **EMPOWERING :** l'infrastructure offre une DevX parfaite aux équipes de développeurs l'utilisant

Comme chez Thoughtworks, ces 4 cadrans sont subdivisés en “rings” dont vous trouverez les définitions à la page suivante.

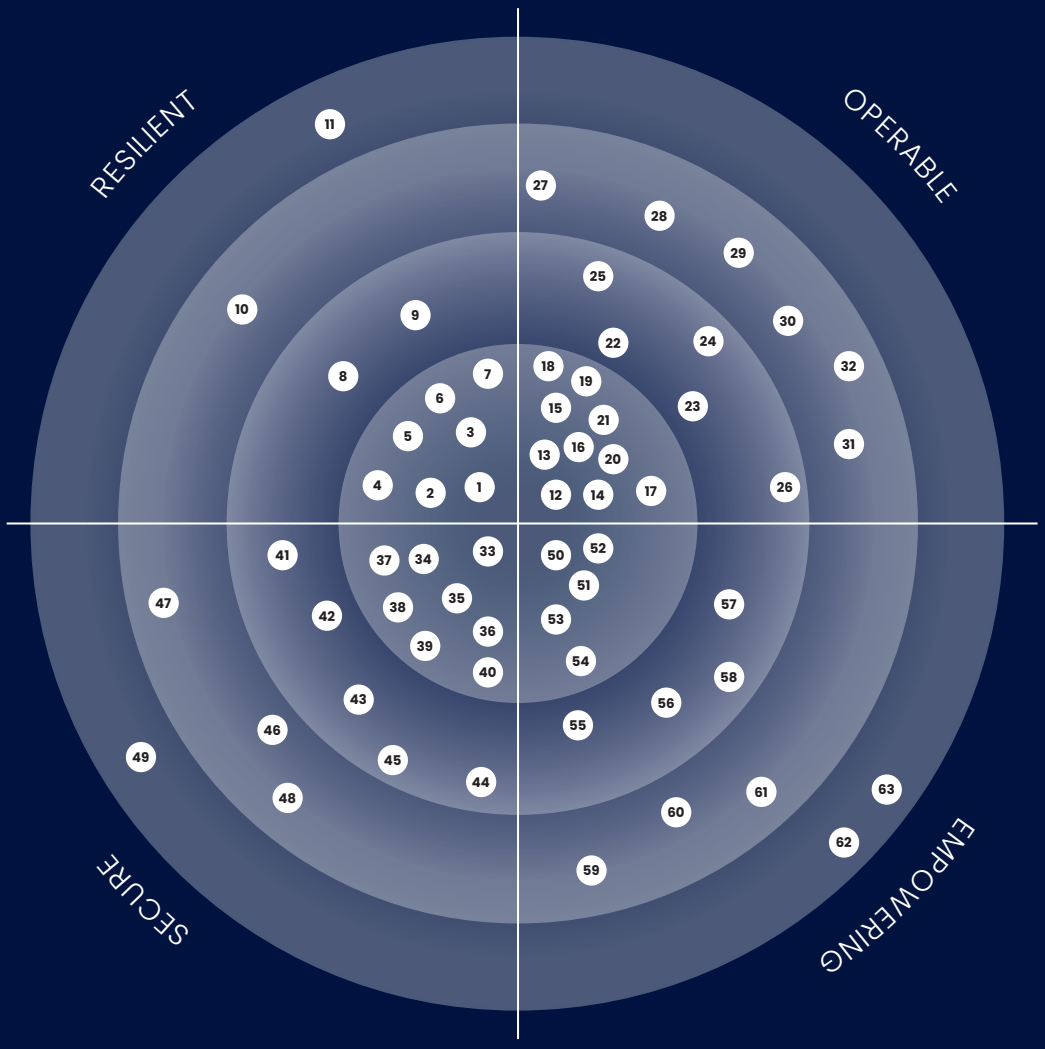
Pour réaliser ce Tech Radar, nous avons listé tous les outils qui nous permettent, ou pour certains nous empêchent, d’atteindre la résilience, l’opérabilité, la sécurité et l’empowerment. Pour chacun de ces outils, nous avons débattu pour répondre à ces questions : a-t-il sa place dans le Tech Radar ? Dans quel Ring le classer ? Et par rapport à l’an dernier, avons-nous un nouvel avis à apporter à la communauté ?

Trois principes ont guidé nos choix éditoriaux :

- **nous avons minimisé le nombre d’outils SaaS** présents car nous avons la conviction que le monde de l’open source nous fournit un niveau de service équivalent à la majorité des alternatives commerciales
- **nous vous proposons uniquement des outils que nous avons utilisés** sur des projets en conditions réelles. Il nous semble impossible d’avoir un avis pertinent sur l’apport d’un outil à la résilience, la scalabilité, la sécurité ou l’expérience développeur s’il n’a pas été éprouvé en production
- **nous n’avons pas mentionné de Cloud Provider**, car le choix d’un provider est extrêmement contextuel et ne se résume pas à un choix technique.

Comment lire le Tech Radar ?

Ce Tech Radar regroupe une soixantaine de technologies Cloud et DevOps éprouvées par les experts de Theodo Cloud durant plus de 5 ans de projets.



Les Blips

Ils sont regroupés d'après notre framework de qualité ROSE (présenté à la page 6) et 4 niveaux d'adoption classés du centre vers l'extérieur du radar :

ADOPT
Nous recommandons l'usage de ces technologies, en cohérence avec votre stack technique.

TRIAL
Ces technos sont prêtes à l'emploi, mais nous pensons qu'elles pourraient encore être optimisées.

ASSESS
Nous vous invitons à vous documenter sur ces technologies si elles répondent à un besoin spécifique.

HOLD
Nous ne sommes pas convaincus par ces technologies et ne les recommandons pas à ce stade.

RESILIENT

1. Architecture ARM
2. CaaS
3. Karpenter
4. KEDA
5. Kubernetes
6. PCA
7. Synthetic Monitoring
8. k6
9. Locust
10. Nomad
11. CloudnativePG

OPERABLE

12. Datadog
13. Grafana
14. KPIs infra (RED Method)
15. Kustomize
16. OpenTelemetry
17. Prometheus operator
18. Renovate
19. Terraform
20. Terragrunt
21. Terraspace
22. Atlantis
23. Burrito
24. Custom Kubernetes Operator
25. Excalidraw +
26. Kubernetes Gateway API
27. Grafana Mimir
28. Crossplane
29. Guacamole
30. OpenTofu
31. Pulumi
32. Terratest

SECURE

33. Checkov
34. External Secrets
35. Helm Secrets
36. Hub & Spoke
37. Kyverno
38. Network Policies
39. SonarQube
40. Vault
41. Architecture Zero-Trust
42. BuildKit
43. Cilium
44. Falco Security
45. Linkerd
46. Boundary
47. Istio
48. OAuth2-proxy
49. Open Policy Agent

EMPOWERING

50. ArgoCD
51. ELK
52. Helm
53. Integrated CI/CD
54. Sentry
55. Github Copilot
56. Loki
57. Platform Engineering
58. Preview Environments
59. Argo Rollout
60. Backstage
61. CI/CD Kube Native
62. Fluxcd
63. Jenkins

Les Tags

Pour cette nouvelle édition, nous avons conçu le Tech Radar non seulement comme un outil de veille technologique, mais également d'aide à la prise de décision.

Grâce aux tags présents sur les pages ci-dessous, nous vous invitons à comparer les technologies aux usages similaires afin de sélectionner les outils les plus appropriés à votre contexte.

#Autoscaling
3. Karpenter
4. KEDA

#Bastion
46. Boundary

#CI/CD
53. Integrated CI/CD
59. Argo Rollout
61. CI/CD Kube Native
63. Jenkins

#DevX
58. Preview Environments
60. Backstage

#LoadTesting
8. k6
9. Locust

#GestionDes Secrets
34. External Secrets
35. Helm Secrets
40. Vault

#GitOps
50. ArgoCD
61. CI/CD Kube Native
62. Fluxcd

#IaC
19. Terraform
20. Terragrunt
21. Terraspace
22. Atlantis
30. OpenTofu
31. Pulumi
32. Terratest

#TACoS
22. Atlantis
23. Burrito

#Networking
36. Hub & Spoke
38. Network Policies
43. Cilium
45. Linkerd
47. Istio

#Observabilité
7. Synthetic Monitoring
12. Datadog
13. Grafana
16. OpenTelemetry
17. Prometheus operator
27. Grafana Mimir
51. ELK
54. Sentry
56. Loki

#Orchestration
2. CaaS
5. Kubernetes
10. Nomad

#PoliciesAsCode
33. Checkov
37. Kyverno
49. Open Policy Agent

#ShiftLeft Security
33. Checkov
39. SonarQube

#ZeroTrust
41. Architecture Zero-Trust
48. OAuth2-proxy

LE CADRAN

Resilient

11 BLIPS | 7 ADOPT | 2 TRIAL | 1 ASSESS | 1 HOLD

La résilience est la capacité à maintenir un niveau de service élevé malgré les aléas du cloud et les éventuelles pannes.

Pour qu'une infrastructure soit résiliente, elle doit répondre efficacement aux enjeux de scalabilité. Nous devons être en mesure de détecter les problèmes et d'être en capacité de rétablir rapidement le service. L'ensemble des outils et concepts de ce cadran sont utilisés par nos Site Reliability Engineers pour répondre à ces enjeux essentiels.
Les technologies et les méthodes choisies sont fréquemment utilisées chez nos clients. Elles nous permettent d'avoir confiance dans les infrastructures Cloud que nous déployons et nous assurent de pouvoir les infogérer correctement.

PAR **THIBAUT ROBINET**
Lead Ops

ADOPT
1/7 blips

1 Architecture ARM

Utiliser l'architecture ARM apporte souvent de meilleures performances pour un coût et une consommation énergétique moindres. Sa démocratisation en a fait un levier FinOps et GreenIT facilement actionnable.

L'architecture ARM (Advanced RISC Machine), **réputée pour sa faible consommation d'énergie et ses performances optimisées**, a longtemps été prisée pour les appareils mobiles. Ces dernières années, ARM a gagné sa place dans les datacenters, remplaçant l'architecture x86 traditionnelle.
AWS promet jusqu'à 20% d'économies et 60% de gain en efficacité énergétique grâce à ses instances Graviton, en comparaison avec des machines équivalentes x86. Cette architecture est donc un levier efficace face à des problématiques de réduction des coûts et de durabilité.

Aujourd'hui, une grande majorité des logiciels sont disponibles via des images Docker ARM, et des outils tels que QEMU et BuildKit simplifient la construction d'images multi-plateformes pour les applications. Dans le cas de services cloud managés (bases de données, FaaS…), **le passage à l'ARM est souvent transparent sur le plan technique, mais se reflète dans les économies réalisées**.
Sur Kubernetes, nous pensons que l'adoption de l'architecture ARM couplée à des outils d'autoscaling performants comme Karpenter permet d'utiliser une large gamme de machines et de s'exposer ainsi à un maximum d'instances spot. Par exemple, pour Bpifrance, nous avons déployé une infrastructure à grande échelle, 100% multi-architecture. Sur cette infrastructure, Karpenter sélectionne les machines spot les plus économiques et les plus disponibles : **ce sont les instances ARM qu'il choisit presque exclusivement**.

NOTRE POINT DE VUE

Nous recommandons l'utilisation de machines ARM, mais seulement après une évaluation des performances spécifiques à chaque charge de travail avant toute transition définitive. Bien qu'elles soient en général plus efficaces, **leur rendement dépend des technologies utilisées et des algorithmes implémentés.**

ADOPT
2/7 blips

2 CaaS

Les services de "Container as a Service" permettent de se rapprocher du "Serverless" sans fondamentalement impacter la façon dont vous organisez ou développez vos applications.

#Orchestration

Aujourd'hui la plupart des Cloud Providers proposent une offre CaaS (par exemple Cloud Run sur GCP, AWS Lambda / ECS sur AWS, ACA sur Azure). **Elle permet un gain important en coûts de maintenance et d'hébergement** de par certaines de leurs features (par exemple, le scale à 0). Utiliser un service CaaS aujourd'hui peut être une première grande étape dans la modernisation de vos applications ou la création d'une nouvelle application.

Le CaaS apparaît comme une vraie alternative à :

- Kubernetes, complexe à mettre en place et à maintenir,
- Serverless, qui implique une transformation architecturale.

Un autre avantage du CaaS est qu'il est par définition **moins "vendor-lock"** que d'autres services d'hébergement. **Votre application est ainsi packagée via un standard du marché** (une image OCI) et peut potentiellement être déployée sur n'importe quel service supportant ces images OCI, comme un futur cluster Kubernetes.

Nous avons pu observer plusieurs comportements via l'utilisation de ces services : soit ils sont complètement adoptés et conviennent parfaitement aux besoins de nos clients, soit ils manquent de customisation et ceux-ci sont naturellement poussés vers l'utilisation de Kubernetes. Par exemple, il y a de cela 1 an, Cloud Run ne supportait pas le fait que le CPU soit “always allocated” et nous ne pouvions pas l’utiliser pour des applications ayant des processus en background. Ce n’est plus une problématique aujourd’hui.

NOTRE POINT DE VUE

Ces services sont devenus incontournables et nous les recommandons. Si l'utilisation d'un Cloud Provider n'est pas une option, il est toujours possible de construire votre propre CaaS en utilisant des technologies open source comme Knative.

ADOPT
3/7 blips

3 Karpenter

Karpenter gère le cycle de vie des nœuds dans un cluster Kube. Il dispose de fonctionnalités pour réduire la taille de son cluster et choisir les instances cloud les moins coûteuses.

#Autoscaling

Dans le monde du node autoscaling pour les services managés Kubernetes, **deux solutions majeures existent aujourd'hui** :

- Kubernetes Cluster Autoscaler (KCA) : déployé manuellement dans le cluster, il est encore très utilisé pour AWS EKS.
- Les technologies gérées par les fournisseurs de Cloud : notamment GKE avec son mode Autopilot, qui repose souvent sur KCA.

Karpenter se positionne en tant qu'alternative de KCA, avec pour vocation de fonctionner avec n'importe quel Cloud Provider. Aujourd'hui, AWS EKS ainsi qu'Azure AKS sont supportés.

Son fonctionnement n'a pas changé depuis que nous l'avons couvert dans la précédente édition de notre Tech Radar : des CRDs (Custom Resource Definitions) permettent de définir les groupes de nœuds et leurs caractéristiques à l'intérieur de Kubernetes (OS, CPU, RAM, zones de disponibilité…).

Karpenter surveille les nouveaux pods qui ne peuvent pas être assignés à un nœud existant. Dès que cela se produit, Karpenter va assigner le pod à un nœud provenant du groupe qui convient à ses contraintes d'orchestration (tolerations, nodeSelector, (anti)affinities…).

Karpenter est open-source et son développement est très actif, avec des fonctionnalités très intéressantes ajoutées à chaque version : consolidation, disruption, mises à jour automatiques de l'image des nœuds… Prenez le temps de lire chaque changelog !

NOTRE POINT DE VUE

C'est la solution que nous déployons chez tous nos clients utilisant EKS. **Karpenter permet systématiquement de baisser le coût de votre cluster.** En le déployant sur Fargate sur EKS, seul le Control Plane doit être mis à jour. **Notre expérience positive nous a donc poussés à le faire évoluer de Trial vers Adopt.**

ADOPT
4/7 blips

4 KEDA

KEDA permet de mettre à l'échelle les applications déployées dans Kubernetes en se basant sur des événements plus riches que des métriques système de base comme le CPU ou la RAM.

#Autoscaling

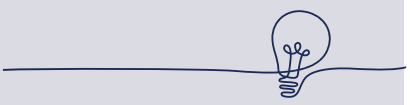
S'assurer que les infrastructures puissent supporter la charge rapidement est un de nos objectifs principaux en tant que SRE. Il est cependant difficile de scaler nos ressources par anticipation car nous nous appuyons généralement sur des métriques système comme la consommation de CPU ou RAM. C'est là que KEDA (**K**ubernetes **E**vent-**D**riven **A**utoscaling) intervient.

KEDA est un composant qui étend les fonctionnalités d'autoscaling de Kubernetes en se basant sur des sources d'événements. Il permet ainsi de scaler nos applications en fonction de la charge événementielle de services comme Kafka, RabbitMQ, AWS SQS, PubSub, pour ne citer qu'eux.

On pourrait donc par exemple scaler des workers RabbitMQ en se basant sur le nombre de messages publiés / seconde dans une queue, ou le nombre de messages en statut "Ready". On peut également imaginer simplement vouloir démarrer le worker lorsque des messages arrivent dans la queue dans le but de faire des économies. Nous avons bien souvent tendance à nous satisfaire du HPA natif de Kubernetes qui permet une mise à l'échelle des applications en se basant sur des métriques telles que le CPU ou la RAM. Notre expérience nous a montré que c'est loin d'être une solution optimale, surtout quand la charge est fluctuante. En permettant une mise à l'échelle automatique basée sur des événements, KEDA offre une réactivité et une flexibilité supérieure aux solutions traditionnelles de scalabilité.

NOTRE POINT DE VUE

KEDA est une solution puissante, facile à déployer et à maintenir. Elle s'intègre avec une multitude de sources d'événements telles que les files d'attente de messages, les bases de données ou encore les métriques personnalisées (avec Prometheus par exemple) qui en fait un choix idéal s'adaptant à bon nombre de cas d'usage.



ADOPT
5/7 blips

5 Kubernetes

Kubernetes est le standard de la communauté pour les déploiements résilients et scalables d'applications conteneurisées, malgré une charge de travail non négligeable.

#Orchestration

Kubernetes est un orchestrateur de conteneurs permettant de déployer à l'échelle un grand nombre d'applications et de garantir leur résilience. En effet, il permet de répondre notamment aux problématiques suivantes :

- Adapter le nombre de réplicas de l'application en fonction de la charge ;
- Assurer la reproductibilité des déploiements ;
- Ajouter des composants externes sans trop complexifier le déploiement.

Avant l'ère des conteneurs, on aurait sûrement utilisé des scripts Bash ou des playbooks Ansible pour résoudre ces problématiques. Tout cet outillage est aujourd'hui remplacé par **des pipelines de CI/CD permettant de produire les images des conteneurs utilisées pour le déploiement dans Kubernetes**.
De plus, le load balancing, qui nécessitait auparavant un système d'adresses IP virtuelles, est aujourd'hui pris en charge :

- Nativement par Kubernetes pour le load balancing interne du cluster ;
- Via des intégrations aux Cloud Providers pour le load balancing externe ;
- Via des extensions comme MetalLB sur une infrastructure bare metal.

Enfin, **la gestion de la capacité de traitement et l'installation d'outils externes sont facilitées via l'utilisation de son interface en YAML** et des différents outils qui en ont émergé comme Kustomize ou Helm. Cependant, il est important de considérer la maintenance de Kubernetes, qui reste conséquente même en utilisant le service managé d'un Cloud Provider, ou une alternative plus visuelle à Kubectl comme K9s. Ainsi, **un CaaS sera certes moins extensible, mais plus léger à maintenir que Kubernetes**.

NOTRE POINT DE VUE

Ainsi, nous recommandons Kubernetes à tous nos clients qui ont la capacité de le maintenir et cherchent à mettre en place une plateforme extensible sur laquelle **déployer facilement des applications à grande échelle.**

ADOPT
6/7 blips

6 PCA

Un Plan de Continuité d'Activité, ou PCA, est une pratique de résilience d'infrastructure, simplifiée par l'aspect distribué des Cloud Providers.

Un Plan de Continuité d'Activité assure la disponibilité d'une infrastructure en cas de désastre.
Il est essentiel pour maintenir un haut niveau de disponibilité et une expérience utilisateur fluide.

Avant de le concevoir, **il faut estimer le Recovery Time Objective (RTO) et le Recovery Point Objective (RPO) de l'application** :

- Le temps de récupération (RTO) est le délai maximal acceptable entre l'interruption de service et la restauration du service.
- Le point de récupération (RPO) est la durée maximale acceptable depuis le dernier point de récupération de données.

Les Clouds Providers publics proposent plusieurs façons de garantir un PCA :

- Les régions AWS sont localisées dans des pays. Il peut y en avoir plusieurs par pays et elles sont sous-divisées en zones de disponibilités.
- Les zones de disponibilités regroupent plusieurs datacenters différents qui hébergent les ressources du Cloud Provider.

En fonction des objectifs, l'application sera hébergée sur :

- Une seule zone de disponibilité au sein d'une région
- Plusieurs zones de disponibilité au sein d'une région, (l'approche la plus commune)
- Plusieurs zones de disponibilité au sein de plusieurs régions. Cela garantit une haute disponibilité, mais c'est le plus difficile à maintenir, car il faut garantir la cohérence des données entre plusieurs régions tout en gardant un temps de réponse bas.

NOTRE POINT DE VUE

Chez Theodo Cloud, nous **recommandons de donner la priorité à la mise en place d'un PCA par rapport à un PRA, Plan de Reprise d'Activité, suite à un désastre.** En effet, ceux-ci sont coûteux à tester, il en découle rarement des leviers activables et nous sommes convaincus qu'il est surtout primordial qu'ils soient effectués à l'échelle du Cloud Provider.

ADOPT
7/7 blips

7 Synthetic Monitoring

Technique de surveillance de vos applications qui consiste à simuler un utilisateur réel avec des robots pour détecter des dysfonctionnements.

#Observabilité

Le Synthetic Monitoring est une technique complémentaire de surveillance de vos applications qui consiste à **simuler un utilisateur réel avec des robots pour détecter des dysfonctionnements**. En effet, la complexité des architectures cloud fortement distribuées peut rendre incomplète une supervision uniquement basée sur la disponibilité des éléments d'infrastructure.

Généralement, on s'attachera à tester en priorité les chemins critiques d'une application. **C'est le parcours utilisateur qui représente la plus forte valeur business**. Par exemple, pour un site de vente en ligne, il s'agira du tunnel d'achat :

- recherche d'un produit
- ajout au panier
- paiement

Ce test permettra de s'assurer que vos clients peuvent réellement acheter sur votre site. Il a aussi le mérite de valider que vos services backend, comme la case recherche, le stockage des sessions, etc. sont tous opérationnels au moment du test.

D'un simple appel à une API backend au parcours à plusieurs étapes, **de nombreux outils vous permettent de mettre en place du Synthetic Monitoring**. Que ce soit des outils payants comme Datadog, NewRelic ou DynaTrace ou bien des outils open source comme Blackbox Exporter (Prometheus Stack), le choix est vaste. Veuillez toutefois noter qu'il est préférable d'**exécuter ces scénarios depuis l'extérieur de l'infrastructure,** afin de vous positionner comme un "vrai" client de votre application et ainsi détecter des dysfonctionnements tels que des coupures ou des

latences du réseau.

Attention cependant aux dépendances externes : elles peuvent générer des alertes sans que vous ne puissiez agir. Néanmoins, il apparaît important d'être informé si l'un de vos fournisseurs est indisponible. Dans ce cas-là, on vous recommandera d'**avoir des procédures spécifiques pour traiter ces évènements** et d'implémenter dans vos applications un système de circuit breaker. Votre application se mettra en maintenance automatiquement et vous pourrez avoir un traitement spécifique lors de vos astreintes. On peut imaginer n'être alerté que si le service est inaccessible plus de 1 heure.
Le monitoring de vos applications doit être un standard dans votre cycle de développement. Comme la sécurité ou la performance, il faut une étape consacrée à la mise en place d'une surveillance du bon fonctionnement de vos services. Ne négligez pas non plus la maintenance et l'évolution. Vous ne devriez jamais passer en production sans avoir des sondes en place pour être averti en cas de dysfonctionnement. Il n'y a rien de plus frustrant que d'être prévenu par vos clients sans vous en être rendu compte avant.

NOTRE POINT DE VUE

On recommandera une **approche "automatique"** en créant des sondes pour chaque application déployée. C'est ce que nous faisons sur nos projets avec la flexibilité de l'API Kubernetes. Ainsi, aucun de nos projets ne passe en production sans avoir le monitoring adéquat validant que le service est bien rendu.

TRIAL
1/2 blips

8 k6

k6 est un framework de load testing extensible, idéal à utiliser en local, dans votre cluster Kubernetes, ou de façon managée via Grafana Cloud.

#LoadTesting

k6 est un framework extensible de load testing développé par Grafana Labs. **Il permet de tester la résilience de votre infrastructure face à des pics de charge sur des parcours critiques**.

k6 utilise **JavaScript** comme langage de scripting. Il est donc facile pour les développeurs web d'être autonomes pour créer des scénarios de tests à lancer contre leur application.

La force de k6 réside dans le système **xk6, qui permet d'ajouter des extensions à l'outil**. Le panel de fonctionnalités additionnelles à votre disposition s'est bien élargi depuis 2023. Dans les nouveaux venus, nous retrouvons par exemple le support de gRPC, de JavaScript Streams API ou encore des fonctionnalités encore plus avancées pour les browser tests. La nouvelle qui nous a le plus réjoui chez Theodo Cloud est **l'arrivée d'un dashboard web intégré nativement à k6** : plus besoin de construire son propre dashboard Grafana pour analyser ses résultats !

k6 se dote aussi d'un opérateur Kubernetes, dont 8 releases sont sorties depuis notre dernier Tech Radar. Entre autres, il existe maintenant un chart Helm officiel afin de faciliter son déploiement et une CRD PrivateLoadZone qui permet à k6 Cloud de lancer les tests depuis votre cluster Kubernetes. Cependant, il est toujours nécessaire de construire sa propre image de runner k6 pour utiliser des extensions, ce qui est regrettable.

Pour de simples tests HTTP, **lancer k6 en local ou depuis une VM sera suffisant** pour des tests sporadiques. L'opérateur Kubernetes, quant à lui, sera votre meilleur allié pour effectuer des tests distribués à plus grande échelle. Enfin, k6 est maintenant complètement intégré à la stack Grafana Cloud, si vous cherchez une offre managée.

NOTRE POINT DE VUE

k6 nous semble aujourd'hui être la **solution de tests de charge la plus prometteuse**, même face à Locust, pour toutes vos infrastructures cloud et Kubernetes.

TRIAL
2/2 blips

9 Locust

Locust est un outil de test de charge de votre application à base de script Python.

#LoadTesting

Locust fait partie de la famille des **outils dits de "Load testing"**, il permet de décrire des scénarios d'usage de vos applications et ensuite de les faire jouer en simulant de nombreux utilisateurs virtuels.

Réaliser ces tests permet de :

- Gagner en confiance sur la tenue en charge de votre application ;
- Vérifier que votre infrastructure actuelle est suffisante pour tenir une charge plus importante ;
- Identifier les différents axes d'amélioration de la performance de votre application.

La force de Locust réside dans sa **simplicité d'utilisation mais également de mise à l'échelle** avec son modèle d'agent central et de workers qui permettent d'atteindre de manière quasi illimitée l'attendu de performance que vous voulez valider.

Les scénarios sont **très simples à écrire** et si vous êtes familier avec **Python** vous n'aurez que très peu de difficultés à écrire vos premiers tests.

Locust vous mettra également entre les mains une **interface utilisateur présentant des dashboards de performance en temps réel** ainsi qu'un contrôle sur les tests en cours (Arrêt/Marche).

L'outil propose régulièrement des améliorations. Il s'est par exemple étoffé dans la gestion des infrastructures de test fortement distribuées et le support de protocoles autres que HTTP, comme gRPC. Il est également maintenant disponible dans une **version cloud qui accélère sa mise en place**.

NOTRE POINT DE VUE

Il existe de nombreux outils de test de charges sur le marché mais Locust est aujourd'hui l'un des plus simples et nous vous conseillons de l'essayer au même titre que k6.

10 Nomad

Nomad est un orchestrateur de tâches.

ASSESS
1/1 blips

#Orchestration

L'orchestrateur de tâches est devenu l'un des piliers de l'infrastructure moderne. Kubernetes étant le plus connu, on peut rapidement se persuader que c'est la seule alternative viable et qu'il faut l'adopter par défaut.

C'est un choix que de nombreuses entreprises font car la gestion de conteneurs avec Kubernetes est très mature. Cependant, **l'infrastructure d'une grande part des entreprises est hétérogène** (conteneurs, VM, webservices…) **et serait très coûteuse à conteneuriser**. Elles peuvent alors se tourner vers Nomad.

Nomad est un orchestrateur de tâches généraliste créé par Hashicorp. En plus de gérer des conteneurs, Nomad est doté de drivers supportant des tâches sous la forme de machines virtuelles, de simples scripts, d'applications Java, etc. Se présentant comme un simple binaire, **Nomad est léger et simple d'installation**.

Adossé à Consul (solution de Service Mesh d'HashiCorp), **il est possible de complètement fédérer ses applications, peu importe la forme qu'elles ont**. Cela procure à Nomad une **bonne extensibilité et une capacité d'adaptation à l'existant que n'a pas Kubernetes**. La migration vers un orchestrateur est donc beaucoup moins coûteuse que s'il fallait conteneuriser toutes ses applications.

NOTRE POINT DE VUE

Nos réserves portent sur son **manque d'intégration avec les Cloud Providers existants** ainsi que son adoption moins grande par la communauté par rapport à Kubernetes. C'est toutefois un outil qu'il faut considérer si vous avez une infrastructure on-premise volumineuse ou que vos ressources à allouer à l'orchestration de vos tâches sont limitées.

HOLD
1/1 blips

11 CloudNativePG

CloudNativePG est un opérateur Kubernetes pour gérer le cycle de vie d'une base de données PostgreSQL.

CloudNativePG utilise 4 CRDs pour gérer des bases de données PostgreSQL :

- **Cluster**, qui définit un cluster PostgreSQL avec une ou plusieurs instances en synchronisation.
- **Pooler**, qui permet de déployer PGBouncer devant un cluster donné.
- **Backup**, qui représente une sauvegarde du cluster à un point donné.
- **ScheduledBackup**, qui permet de créer des objets Backup selon une fréquence donnée.

En s'appuyant sur des fonctionnalités mises à disposition par les Cloud Providers et Kubernetes, telles que les volumes Block Storage, les Volume Snapshots et l'Object Storage, **il est possible de créer des bases de données avec un bon niveau d'opérabilité**. Accompagné d'une connaissance de PostgreSQL et ses options de configuration, CloudNativePG couvrira les besoins de vos applications.

Cependant notre expérience avec cet opérateur en particulier nous pousse à mettre en garde contre son utilisation en production si votre usage de PostgreSQL est intensif (téraoctets de données, milliers de transactions par seconde) :

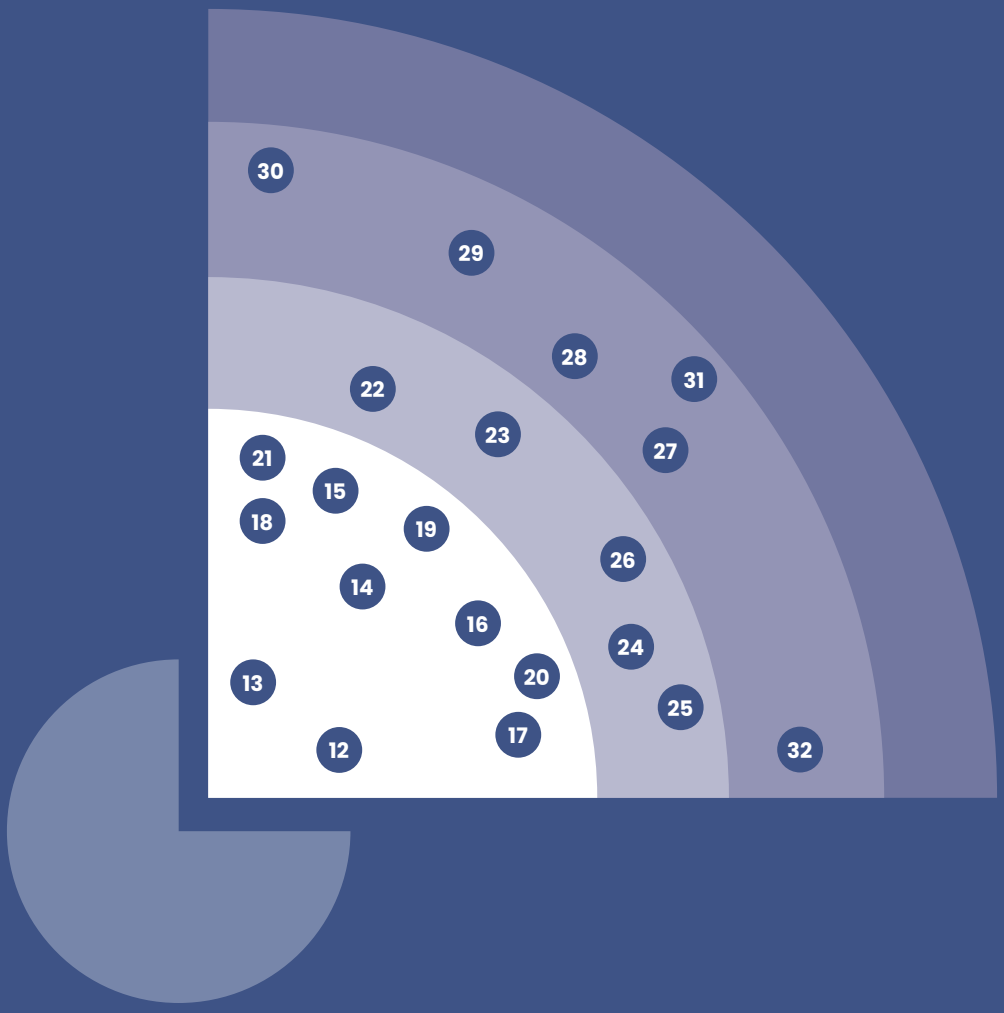
- Une mise à jour majeure de PostgreSQL impose de créer un nouveau cluster et d'y répliquer les données de l'ancien, obligeant donc un downtime applicatif.
- La réplication d'une base de données non gérée par CloudNativePG est possible, mais mal documentée.
- Un RBAC Kubernetes trop permissif permet à quelqu'un de supprimer votre base de données en une action.

NOTRE POINT DE VUE

Tout cela s'accompagne donc de coûts de maintenance et d'opération non négligeables pour garantir la bonne préservation de vos données. Un service managé de base de données vous permettra d'atteindre une meilleure sérénité à moindre coût.

LE CADRAN

Operable



21 BLIPS | 10 ADOPT | 5 TRIAL | 6 ASSESS | 0 HOLD

Pour ce cadran “Operable”, nous avons choisi de présenter des outils et des technologies qui permettent d’améliorer les opérations courantes sur les infrastructures Cloud Native.

Cela inclut des technologies pour le monitoring, la journalisation, la gestion des mises à jour et la gestion de l’Infrastructure As Code.

Une infrastructure opérable est essentielle pour garantir des applications et des services fiables et qu’en cas d’incident, le temps de résolution (MTTR) soit le plus faible possible.

Nous avons cette année décidé d’inclure certains outils SaaS que nous considérons comme standard du marché, notamment ceux où les outils open-source ne permettent pas d’atteindre leur niveau de service sans un investissement conséquent.

PAR **GUILLAUME LECCESE**
Lead Ops

ADOPT
1/10 blips

12 Datadog

Datadog est un SaaS payant d'observabilité des infrastructures Cloud. Il surveille la performance et la santé des applications, permettant de détecter et résoudre rapidement les problèmes.

#Observabilité

Leader du marché* dans le domaine de l'observabilité, **Datadog propose un catalogue complet de solutions d'observabilité, de monitoring et d'alerting**. Il facilite la détection et la résolution rapide des problèmes, améliore la collaboration inter-équipes et optimise l'utilisation des ressources pour les entreprises de toutes tailles.

Datadog offre une solution de monitoring complète en intégrant à la fois les approches whitebox et blackbox :

- En **whitebox**, Datadog permet une surveillance granulaire des métriques internes des applications et des infrastructures, facilitant la détection précoce des anomalies et l'optimisation des performances.
- En **blackbox**, Datadog simule les interactions des utilisateurs finaux pour identifier les problèmes de disponibilité et de performance externe, garantissant ainsi une expérience utilisateur optimale.

Cette double approche assure une couverture exhaustive, permettant de détecter et de résoudre les problèmes à tous les niveaux de votre stack technologique.

Nous adoptons Datadog pour sa facilité d'utilisation, son provider Terraform, qui nous permet d'automatiser notre gestion de l'alerting par client et de la maintenir facilement, et enfin pour son offre de monitoring qui s'adapte à toutes les infrastructures et technologies applicatives.

*source : Gartner

NOTRE POINT DE VUE

Datadog est un outil simple d'utilisation avec une interface intuitive, offrant de nombreuses intégrations permettant de s'adapter aux différentes infrastructures. Si toutefois l'outil semble trop cher, vous pouvez vous tourner vers des solutions d'observabilité open source telles que Prometheus ou la stack ELK, qui nécessitent néanmoins plus de configuration et de maintenance.

ADOPT
2/10 blips

13 Grafana

Grafana est une plateforme de visualisation open source pour tous vos environnements cloud.

#Observabilité

Grafana est un outil de visualisation open source créé par l'entreprise éponyme, Grafana Labs. Il permet aux utilisateurs de **créer des dashboards dynamiques et personnalisables** pour surveiller et analyser les metrics liées à votre infrastructure. **Grafana prend en charge une large gamme de sources de données**, permettant aux utilisateurs de centraliser et de visualiser leurs données à partir de divers systèmes. Voici quelques-unes des sources de données les plus populaires prises en charge par Grafana :

- Prometheus : Pour la surveillance des systèmes et des services.
- Elasticsearch : Pour la recherche et l'analyse de journaux.
- InfluxDB : Pour le stockage et l'analyse de séries temporelles.
- Loki : Pour la surveillance des journaux avec une approche scalée.
- Base de données (MySQL, Postgres, TimescaleDB, ...)
- Cloud provider (AWS Cloudwatch, GCP Monitoring, Azure Monitoring, ...)

Son interface utilisateur intuitive vous permet ensuite de regrouper les différentes données sous forme de graphiques en temps réel, jauges, diagrammes en barre pour ne citer que ceux-là.

Grafana est un incontournable du monitoring pour vos clusters Kubernetes, très facile à installer et configurer grâce à son chart Helm. Si vous ne voulez pas avoir la responsabilité de gérer votre outil de visualisation, pas de soucis, il existe une offre SaaS Grafana Cloud.

NOTRE POINT DE VUE

Avec l'essor des architectures en microservices et de l'utilisation du cloud pour créer des environnements complets et complexes, nous recommandons Grafana à tous nos clients qui veulent une alternative open source à Datadog, que ce soit pour les équipes opérationnelles ou les équipes de développement.

ADOPT
3/10 blips

14 KPIs d'infrastructure (RED Method)

La méthode RED optimise la surveillance des KPIs de l'infrastructure pour ne conserver que les plus critiques afin d'améliorer les performances.

La méthode RED (Rate, Errors, Duration) est une **approche pour surveiller et améliorer les performances des infrastructures**. Elle repose sur trois KPIs essentiels pour évaluer l'efficacité des systèmes : le taux (Rate), les erreurs (Errors) et la durée (Duration).

Le taux mesure le nombre de requêtes ou d'opérations traitées par le système sur une période donnée. Ce KPI est crucial pour **comprendre la charge de travail que le système peut supporter** et pour identifier les périodes de pic d'activité. L'analyser permet de prévoir les besoins en ressources et d'**ajuster la capacité en conséquence**.

Les erreurs quantifient le nombre de requêtes qui échouent ou de problèmes rencontrés par le système. Suivre ce KPI est vital pour **maintenir la fiabilité et la stabilité des infrastructures**. Une augmentation des erreurs peut indiquer des failles dans le système, des problèmes de configuration ou des défaillances matérielles. Identifier et résoudre rapidement ces erreurs permet de minimiser les **interruptions de service et d'améliorer l'expérience utilisateur**.

La durée mesure le temps moyen nécessaire pour traiter une requête ou une opération. Ce KPI permet d'**évaluer la performance et l'efficacité du système**. Un temps de traitement plus long peut signaler des goulets d'étranglement ou des inefficacités dans les processus. En optimisant la durée, on peut **réduire la latence et améliorer la réactivité des applications**, ce qui est crucial pour les utilisateurs finaux.

NOTRE POINT DE VUE

En combinant ces trois indicateurs, la méthode RED offre une vue d'ensemble des performances des infrastructures, permettant aux équipes techniques de détecter rapidement les problèmes, de planifier les améliorations et d'assurer une performance optimale : un outil indispensable pour une **gestion proactive.**

ADOPT
4/10 blips

15 Kustomize

Kustomize simplifie la gestion des déploiements complexes dans Kubernetes.

Kustomize permet de **gérer les configurations des ressources pour Kubernetes** via des fichiers YAML, **avec une syntaxe facile à utiliser**. Il permet également de gérer des configurations complexes pour des applications multi-environnements en appliquant des configurations en série.

En effet, **Kustomize applique des patchs et overlays** sur une configuration de base. Cela permet de gérer des configurations complexes plus simplement qu'avec Helm où il faut redéfinir un fichier de valeurs à chaque fois. Le tout en gardant votre code aussi DRY ("Don't Repeat Yourself") que possible !

De plus, Kustomize génère automatiquement de nouveaux secrets Kubernetes chaque fois que vous modifiez un champ de données (data field) dans le fichier de configuration. Cette fonctionnalité permet de **maintenir la sécurité des secrets tout en simplifiant vos rollbacks**.

Cependant, Kustomize est plus difficile à appréhender pour des personnes n'étant pas familières avec les ressources Kubernetes et leur déclaration en YAML. C'est aussi un outil moins flexible que Helm en termes de personnalisation : il n'est pas possible d'intégrer de logique via du templating par exemple.

Nous pensons que Kustomize et Helm sont deux outils complémentaires pour la gestion des configurations de déploiement pour Kubernetes. Helm est idéal pour gérer des packages d'application complexes, tandis que Kustomize est parfait pour personnaliser des configurations spécifiques pour différents environnements de déploiement.

NOTRE POINT DE VUE

Kustomize est donc un outil puissant pour la gestion des configurations de déploiement dans Kubernetes, avec une syntaxe simple qui complète parfaitement Helm.

ADOPT
5/10 blips

16 OpenTelemetry

OpenTelemetry et son collecteur facilitent la collection de métriques, logs et traces au moyen de standards et de configuration unifiée.

#Observabilité

À l'ère du Cloud et de Kubernetes, les infrastructures qui font vivre les services que l'on utilise aujourd'hui sont bien souvent distribuées. **Les bénéfices en termes de scalabilité et de résilience ne sont plus à démontrer, mais cela peut nuire à la compréhension globale du système**.

Il est donc indispensable de mettre en place un bon niveau d'observabilité.
Il existe 3 grands types de données :

- **Les métriques**, pour créer des dashboards présentant les données numériques générées par l'infrastructure ;
- **Les logs**, pour analyser le comportement des composants qui traitent l'information ;
- **Les traces**, pour pouvoir suivre le parcours des requêtes au travers de l'infrastructure et ses composants.

OpenTelemetry a été créé pour faciliter la mise en place de cette observabilité de manière unifiée grâce à ses librairies et outils. L'intérêt d'OpenTelemetry réside dans la génération, la collection et l'export de données d'observabilité vers n'importe quel backend tel que Jaeger, Datadog, NewRelic ou encore Prometheus.

OpenTelemetry propose notamment des SDKs dans les langages de programmation les plus utilisés. **Cela permet aux développeurs de faire remonter des données qui peuvent être très utiles au debug**. Cependant, ce qui nous intéresse ici est le collecteur OpenTelemetry.

Le collecteur se déploie sous forme de process ou container (sur une VM, ou dans Kubernetes) **et fait le pont entre l'infrastructure et les backends d'observabilité.** Pour chaque type de donnée, il est possible de définir source(s), transformation(s) et destination(s) et de les combiner dans un système intuitif de pipelines.

NOTRE POINT DE VUE

Il est intéressant de noter que la documentation de certains composants peut être cachée dans des repositories GitHub qui ne sont pas faciles à trouver. Cependant, la facilité d'opération et la forte activité autour du développement d'OpenTelemetry nous a convaincus de le placer dans le ring Adopt.

ADOPT
6/10 blips

17 Prometheus Operator

Prometheus Operator déploie et gère la stack technique autour de Prometheus afin de monitorer un cluster Kubernetes.

#Observabilité

Prometheus est l'outil de référence pour la **métrologie sur les architectures Kubernetes**. Le déploiement et la gestion sont particulièrement simplifiés grâce à Prometheus Operator tout en ajoutant d'autres composants annexes, mais non moins nécessaires, qui permettent d'améliorer l'opérabilité de votre plateforme :

- alerting avec AlertManager
- monitoring HTTP avec Blackbox Exporter
- visualisation des métriques avec Grafana

L'installation de Prometheus dans votre cluster Kubernetes se fait en une seule commande. Il est possible d'avoir accès à toutes les métriques de votre cluster en quelques minutes et de commencer vos premières analyses grâce aux tableaux de bord génériques installés dans Grafana. L'opérateur Prometheus permet de déclarer dans l'API Kubernetes votre configuration Prometheus via les CRDs (Custom Resource Definition).

Les CRDs vous offrent la possibilité d'automatiser le déploiement du monitoring, en ajoutant la configuration Prometheus dans vos charts Helm. Ainsi, vos applications pourront être monitorées par défaut, sans avoir besoin d'éditer la configuration de Prometheus.

Toutefois, le problème majeur de Prometheus n'est pas réglé par l'opérateur : proposer une vue consolidée et centralisée dans des architectures multi-environnements et multi-clusters. Vous devrez alors déployer des composants qui permettent de faire le lien entre les différents déploiements : un Grafana central et des solutions comme Thanos pour augmenter la rétention des données.

NOTRE POINT DE VUE

Même si d'autres solutions existent, comme Datadog (payant), Prometheus reste le standard de facto de la communauté pour Kubernetes et demeure un bon choix pour opérer vos clusters. L'opérateur apporte un atout supplémentaire : il vous permet d'industrialiser le monitoring.

ADOPT
7/10 blips

18 Renovate

Renovate permet d'automatiser la mise à jour (Patch Management) des dépendances externes (librairies) de l'infrastructure et des applications.

Le Patch Management est un enjeu crucial pour la sécurité d'une plateforme. Nos infrastructures et applications, utilisant des composants externes souvent open source, nécessitent des mises à jour régulières pour corriger failles de sécurité et bugs. Avec le rythme rapide des mises à jour et l'augmentation des dépendances, tout maintenir à jour est un défi.

Renovate est un outil open source de gestion de dépendances qui automatise la mise à jour des packages dans vos projets. Il analyse vos fichiers de configuration de dépendances (tels que package.json, pom.xml ou build.gradle) et **génère automatiquement des pull requests** pour les mises à jour de packages nécessaires. Renovate est **compatible avec une variété de gestionnaires de packages**, notamment npm, yarn, pip, Maven et NuGet, ce qui le rend facilement adaptable aux différents langages de programmation. Il permet aussi d'analyser les dépendances de votre infrastructure en supportant les charts Helm, les images Docker et les modules Terraform. Vous pouvez l'utiliser avec les Git Provider majeurs.

Hautement configurable, il s'adapte parfaitement aux workflows de développement de nos projets. Il s'intègre via des tâches de CI ou en tant que cronjob dans un cluster Kubernetes afin d'optimiser les traitements avec du cache dans Redis. Ce sera un mode de déploiement qui permettra de scale plus facilement en déployant plusieurs "instances" de Renovate (cronjob Kube) pour répartir la charge et adapter son fonctionnement. Avec une **exécution journalière et le merge automatique des changements** (quand la CI est valide), nous automatisons une partie de la correction des failles de sécurité en appliquant automatiquement les patchs.

Renovate permet de suivre les évolutions de nos dépendances à prendre en compte pour les maintenir à jour (via la liste des Merge Request/Pull Request ouvertes).

NOTRE POINT DE VUE

Renovate nous permet de **réduire le toil du Patch Management** et de dégager du temps pour travailler sur des améliorations qui apporteront plus de valeur au business de nos clients.

ADOPT
8/10 blips

19 Terraform

Aujourd'hui Terraform est l'outil d'IaC qui domine le marché, même si Opentofu n'est pas loin. Il permet de provisionner et de manager des ressources sur tous les Cloud Providers.

#IaC

Nous avons créé des centaines d'infrastructures sur plusieurs Clouds Providers et nous avons utilisé à chaque fois Terraform. Un outil d'IaC est essentiel pour se lancer dans le cloud, car il facilite la collaboration, mais aussi l'opérabilité d'une infrastructure.

Terraform rayonne par différentes fonctionnalités qu'il propose :

- L'utilisation de modules qui permettent de définir des ensembles de ressources qui répondent à un besoin précis et de pouvoir les réutiliser facilement.
- Le state Terraform qui permet de suivre le cycle de vie de chaque ressource.
- La compatibilité avec plusieurs clouds et systèmes grâce aux providers. Par ex : AWS, OVH ou encore Github.

On sait que des outils IaC sont proposés par les clouds providers comme : CDK AWS, cloudformation ou même ARM pour Azure. Mais leur vendor lock-in et leur manque d'interopérabilité nous ont fait pencher pour Terraform. Même s'il est le leader dans l'IaC pour gérer une infrastructure, **il est nécessaire d'avoir un cadre pour la code base**. Nous avons convergé chez Theodo Cloud sur un pattern WYSIWYG (What You See Is What You Get) qui aide à l'uniformisation du code et la collaboration*.

Les points à retenir sont :

- Organiser les states Terraform par rapport à un besoin métier.
- Créer des modules qui répondent à un besoin complexe ou reproductible.
- Ne pas hésiter à factoriser le code quand l'infrastructure évolue.

Terraform est **l'outil de référence** aujourd'hui pour construire et maintenir une infrastructure dans le cloud. Des outils tels que **Terragrunt** améliorent encore davantage sa capacité à gérer l'infrastructure at scale en proposant des fonctionnalités permettant d'éviter la redondance de code, appelée DRY (Don't Repeat Yourself).

TIPS POUR SON UTILISATION

Il ne faut pas négliger les outils de qualité syntaxique et de maintenabilité : terraform fmt, tfllint, tfautomv ou terraform-docs et guacamole.

* https://padok-team.github.io/docs-terraform-guidelines

ADOPT
9/10 blips

20 Terragrunt

Terragrunt est un outil à combiner avec Terraform qui vous aide à améliorer la maintenabilité et la scalabilité de vos codebases.

#IaC

Terraform est le standard actuel de la communauté pour le déploiement as code de ressources cloud. Il dispose notamment de librairies pour la quasi-totalité des ressources des grands Cloud Providers. Cependant, des limitations qui lui sont propres pénalisent les équipes pour **gérer une infrastructure à plusieurs, ou pour gérer de grosses infrastructures**. Dans ce genre de cas, il est souvent nécessaire de découper le déploiement de Terraform en plusieurs modules (parfois appelés layers), afin de les simplifier ou d'éviter la concurrence lors de la modification d'un même state Terraform. Or, ceci est vite **très difficile à gérer** car il devient nécessaire :

- D'utiliser des remote states pour partager les outputs entre states ;
- De dupliquer ou sur-templatiser les configurations de backends qui ne sont pas nativement configurable dans Terraform ;
- De gérer les variables communes ou spécifiques des environnements via des fichiers et des liens symboliques.

Terragrunt vient se placer au-dessus, afin de créer et chapeauter des states Terraform autogénérés. On peut ainsi profiter des fonctionnalités accrues de Terragrunt pour lier les apply (layers) entre eux. Terragrunt permet donc d'avoir un **meilleur lien entre layers**, tout en se reposant ensuite sur les capacités reconnues de Terraform pour le déploiement.

Terragrunt utilise le même langage que Terraform, l'HashiCorp Configuration Language (HCL), avec des fonctionnalités supplémentaires pour l'utilisation à grande échelle. Cela facilite la formation des équipes et limite la sensation d'avoir un nouvel outil à maîtriser. L'utilisation de Terragrunt pour déployer des infrastructures en production nous a inspiré l'écriture de bonnes pratiques, dont le Context Pattern*.

NOTRE POINT DE VUE

D'autres outils essaient aujourd'hui de remplir ce besoin, mais Terragrunt a notre préférence car il parvient au résultat sans trop modifier l'utilisation de Terraform et en ajoutant des fonctionnalités puissantes.

*https://padok-team.Github.io/docs-terraform-guidelines/terragrunt/context_pattern.html

ADOPT
10/10 blips

21 Terraspace

Terraspace est un framework opinionné, orienté DRY et Layering pour démarrer rapidement un projet Terraform avec de bonnes pratiques et une structure recommandée.

#IaC

Terraspace est un framework DRY (Don't Repeat Yourself) conçu pour étendre les capacités de Terraform, l'outil d'IaC incontournable. L'un des principaux avantages de Terraspace est sa capacité à **simplifier et à organiser le déploiement de ressources cloud à grande échelle**. Alors que Terraform est puissant, il présente certaines limitations lorsqu'il s'agit de gérer de grandes infrastructures ou de travailler en équipe. Terraform nécessite souvent une gestion manuelle des modules et des configurations, ce qui peut entraîner des complexités.

Terraspace adresse ces limitations en fournissant une **structure de projet claire et des conventions qui favorisent la réutilisation du code**. Il introduit le concept de "stacks" et de "layers" pour regrouper et gérer les ressources liées telles que les environnements ou le cloud multi-régions. Il facilite la gestion des secrets grâce à des **intégrations avec les gestionnaires de secrets** comme ceux d'AWS, Google ou Azure. Il propose également un fichier Terrafile qui permet de **gérer simplement les dépendances des modules Terraform** de manière centralisée.

Le HashiCorp Configuration Language (HCL) utilisé par Terraform est enrichi par Terraspace. Cela permet, de manière tout à fait optionnelle - rassurez-vous - l'**utilisation de Ruby pour écrire des configurations dynamiques** ou calculer une valeur ici et là.

En termes de formation des équipes, Terraspace offre une courbe d'apprentissage plus douce grâce à ses générateurs de code et à sa structure de projet cohérente et plutôt, il est vrai, "opinionnée". Les développeurs peuvent **rapidement monter en compétences** et contribuer efficacement.

De plus, les conventions DRY de Terraspace réduisent les erreurs et améliorent la maintenabilité du code.

NOTRE POINT DE VUE

Comme d'autres outils tels que Pulumi, Terraspace offre une bonne flexibilité grâce à son intégration avec un langage de programmation (Ruby), tout en maintenant une simplicité et une organisation structurée pour les grandes équipes.

TRIAL
1/5 blips

22 Atlantis

Atlantis permet d'automatiser l'utilisation de Terraform via des pull requests. Elle permet d'avoir un workflow qui maintient la cohérence d'une infrastructure définie en IaC.

#IaC #TACoS

Atlantis est un outil open source conçu pour **automatiser les contributions à une code base Terraform. Il permet d'exécuter les commandes *plan*, *apply* et *import* directement dans la pull request**, offrant ainsi un retour visible immédiatement en commentaire. C'est une application qui peut être hébergée n'importe où et qui utilise le système de webhook de GitHub, GitLab ou Bitbucket.

Cela permet de résoudre les problématiques de collaboration sur des grandes infrastructures, mais aussi d'avoir un historique des modifications effectuées.

Des **workflows plus complexes permettent d'accélérer d'autant plus la DevX** (Developer Experience) :

- Autoplanning à chaque nouveau commit ou pull request, qui permet rapidement d'avoir un état de l'infrastructure et de valider l'impact des changements apportés
- Automerging pour merge la pull request si tous les plans sont fonctionnels

Malgré ses avantages, Atlantis présente encore certaines limites pour devenir une référence :

- Séparation des accès : le serveur qui gère Atlantis conserve les informations d'identification pour accéder à l'infrastructure. Pour gérer plusieurs infrastructures avec des droits distincts, il faut instancier plusieurs serveurs, ce qui peut rapidement compliquer la gestion.
- Scalabilité : l'architecture actuelle d'Atlantis limite sa capacité de mise à l'échelle. Pour des infrastructures de grande envergure, des solutions plus robustes et matures peuvent être préférables.

NOTRE POINT DE VUE

Atlantis est un outil prometteur, mais sa gestion complexe des droits et ses limites de scalabilité sont les raisons pour lesquelles nous le positionnons en "Trial". Des fonctionnalités intéressantes, telles que la détection de drift, sont prévues dans sa roadmap, ce qui en fait un outil à surveiller.

TRIAL
2/5 blips

23 Burrito

Burrito est un TACoS (Terraform Automation and COllaboration Software). Il gère le code Terraform à grande échelle avec une approche GitOps et orientée Kubernetes.

#TACoS

Terraform est massivement utilisé pour la gestion des infrastructures cloud, mais son utilisation à grande échelle pose les problèmes suivants :

- Difficulté à proposer un flux de contribution efficace, incluant la revue de code, les tests et les aperçus sur une pull request/merge request.
- Complexité à monitorer les modifications effectuées sur chaque layer.
- Difficultés à détecter les modifications apportées à l'infrastructure en dehors du code et qui désynchronisent le code avec l'existant.

Burrito se connecte au dépôt de code et propose une interface centralisée. Il permet d'**automatiser les workflows** pour le code IaC tout en **détectant les désynchronisations avec l'existant** grâce à des vérifications régulières et automatiques. Il fournit aussi **un tableau de bord avec un historique des changements et des informations supplémentaires** (coût de l'infrastructure, vulnérabilités, bonnes pratiques…).

Les leaders dans ce domaine incluent Spacelift, Terraform Cloud, Scalr et env0. Bien qu'efficaces, ces solutions ont un coût d'entrée élevé, d'environ 200 euros par mois, leur version gratuite ne suffisant pas à gérer efficacement de grandes infrastructures.

Burrito est une solution open source orientée Kubernetes, développée par Theodo Cloud, visant à être ArgoCD pour Terraform. Burrito fonctionne avec du code Terraform & Terragrunt. Ses fonctionnalités principales sont :

- Visualiser un aperçu des modifications d'une PR/MR sur GitHub / GitLab ;
- Automatiquement appliquer le code d'infrastructure définissant le dépôt de code comme source de vérité (GitOps) ;
- Visualiser l'état de son code via son interface.

NOTRE POINT DE VUE

Cette solution, déjà implémentée chez certains de nos clients, facilite et automatise la gestion de code Terraform sur de grandes infrastructures cloud à moindre coût en capitalisant sur un cluster Kubernetes.

TRIAL
3/5 blips

24 Custom Kubernetes Operator

Les opérateurs custom permettent d'automatiser des tâches dans Kubernetes en ajoutant des fonctionnalités à son API.

Kubernetes est très populaire en tant qu'orchestrateur de containers. Mais c'est avant tout une API extensible. **Créer vos propres opérateurs permet d'ajouter de nouvelles ressources à l'API de Kubernetes et d'en étendre les fonctionnalités**.

En créant votre opérateur, vous allez définir des Custom Resource Definitions (CRDs). Le code de l'opérateur tire parti du pattern de réconciliation de Kubernetes afin de déclencher des événements dans votre cluster à chaque ajout, modification ou suppression d'une instance de CRD. Cela peut aider à automatiser des tâches répétitives (réduire votre toil), et à **ajouter des fonctionnalités personnalisées à des applications**. Si vous vous servez de Kubernetes, vous utilisez sans doute quotidiennement des opérateurs custom comme cert-manager ou ArgoCD.

Dans le cadre d'un SaaS par exemple, chaque nouveau client nécessite la création d'un nouveau tenant. Un opérateur permet d'automatiser la création de toutes les ressources nécessaires juste en déclarant un nouvel objet dans votre cluster Kubernetes !

Cependant, la création d'un opérateur peut s'avérer compliquée : **il faut maîtriser le fonctionnement de Kubernetes ainsi que le cycle de vie et différents edge cases** de ce que vous voulez automatiser. Tester tous les cas d'application n'est pas une mince affaire.

Il est important de noter que vous pouvez **écrire vos opérateurs dans n'importe quel langage de programmation** : Java, Rust… et même Ansible. De notre côté, nous recommandons l'utilisation de Golang : vous trouverez énormément de ressources pour vous aider et l'opérateur-sdk de Red Hat permet de bootstrapper votre code très efficacement.

NOTRE POINT DE VUE

La création d'un opérateur peut offrir de nombreux avantages aux équipes DevOps travaillant avec Kubernetes. Cependant, cela peut également être complexe et nécessiter une certaine expertise en programmation et en Kubernetes.

TRIAL
4/5 blips

25 Excalidraw+

Excalidraw+ est l'édition Premium d'un outil de whiteboarding SaaS qui peut servir à la fois aux techs et aux biz pour partager des schémas.

Excalidraw+ permet d'**obtenir en 2 clics une page blanche sans limite** ("Scène") sur laquelle on peut dessiner des formes comme sur un tableau. Les scènes sont regroupées en "Collections" auxquelles on peut donner des droits par équipes, au sein d'un workspace.

Le gros point fort d'Excalidraw+ est dans sa simplicité. Seules des formes élémentaires (ex : rectangles, cercles, flèches, zones de texte) et une capacité de mise en forme restreinte (ex : couleurs, 4 tailles de police) sont possibles dans l'affichage par défaut. L'utilisateur aventurier pourra ajouter des formes provenant de librairies collaboratives, telles que des logos de services Cloud pour illustrer ses schémas d'architecture.

Cela favorise une **meilleure collaboration quotidienne,** grâce à des représentations graphiques nombreuses et des **schémas d'architecture toujours à jour**, car l'effort à produire pour les maintenir est minimisé.

Excalidraw+ permet de faire tout type de schéma, et de collaborer efficacement à distance avec un support visuel. Si vous avez besoin de faire un schéma et ne savez pas où le faire, il n'y a donc pas à hésiter. Vous pouvez essayer la version gratuite sur excalidraw.com. Chez Theodo Cloud, l'utilisation était majoritairement faite par les équipes tech pour réaliser des schémas d'architecture. Nous avons depuis vu les équipes business se saisir de l'outil pour profiter de ses capacités à tout schématiser facilement.

Comme pour la dernière édition, nous souhaitons signaler les limitations suivantes :

- Gestion des droits trop large, par exemple il faut être administrateur pour pouvoir gérer les droits, ou on ne peut donner des droits sur des dossiers qu'à des équipes
- Un utilisateur ne peut être membre que de 6 équipes à la fois
- On ne peut pas créer de sous dossiers
- Absence de SLA affichés (même si l'application est globalement toujours disponible)
- Impossibilité de choisir la localisation du stockage des données

Nous maintenons donc Excalidraw+ en Trial.

TRIAL
5/5 blips

26 Kubernetes Gateway API

Kubernetes Gateway API gère l'accès aux services Kubernetes depuis l'extérieur du cluster.

Lorsque l'on veut **exposer des services Kubernetes à l'extérieur de notre cluster**, notre réflexe est d'utiliser les **ressources de type Ingress**. Nous déployons donc des Ingress Controller comme ceux de Nginx, Traefik ou encore Kong qui vont avoir leurs propres annotations pour diriger le trafic et gérer les Ingress rattachés à eux. Généralement, les développeurs ainsi que les Ops en charge du cluster travaillent sur ces mêmes ressources, créant parfois des perturbations.

Afin de mieux séparer le rôle de chacun pour gérer l'exposition des services applicatifs, un nouveau concept est apparu depuis peu : Kubernetes Gateway API. Il permet aux Ops de mettre en place une **gateway globale au niveau du cluster** (cross-namespace) et qui a pour point d'entrée un load balancer de type L4 ou L7.

Les développeurs sont alors autonomes pour créer leurs propres HTTPRoutes dans leurs namespaces. À noter que ces ressources apportent nativement plus de fonctionnalités, comme le header-based matching ou le traffic weighting.

Kubernetes Gateway API est encore relativement nouvelle, mais gagne en popularité en raison de sa capacité à simplifier la gestion des routes dans les environnements Kubernetes complexes. Elle offre aussi une meilleure visibilité et un meilleur contrôle sur les gateways, facilitant la détection des erreurs et des problèmes de sécurité.

NOTRE POINT DE VUE

Chez Theodo Cloud, nous trouvons **cette technologie très prometteuse** pour les équipes qui veulent simplifier la gestion des routes dans les environnements Kubernetes. À mesure que cette technologie évolue, elle pourrait devenir la référence et remplacer les Ingress.

ASSESS
1/6 blips

27 Grafana Mimir

Grafana Mimir implémente une API Prometheus et propose nativement de stocker vos métriques sur un bucket.

#Observabilité

Prometheus est le standard open-source pour la collection de métriques d'infrastructure. Son API est implémentée par de nombreux composants que vous avez sûrement déjà déployés dans votre infrastructure.

Certaines contraintes d'un environnement cloud-natif rendent l'utilisation de Prometheus difficile dans certaines situations :

- Aucun moyen de stocker ses métriques en dehors d'un disque attaché à Prometheus
- Prometheus récupère les métriques de vos services; l'inverse est déconseillé par ses mainteneurs
- Pas de multi-tenancy pour cloisonner vos métriques en fonction de leur émetteur

Grafana Mimir implémente une API Prometheus et intègre tous les éléments ci-dessus. Accompagné d'une autre technologie telle que le collecteur OpenTelemetry pour récupérer les métriques et les lui envoyer, il est possible de construire un puits centralisé de données pour toute votre infrastructure.

L'architecture de Mimir est distribuée par nature : chaque fonction (ingestion, query, compactor…) dispose de son microservice que l'on peut mettre à l'échelle indépendamment lorsqu'on le déploie sur Kubernetes. **Il s'intègre naturellement bien avec Grafana pour créer une stack d'observabilité cohérente et opérant sur les mêmes standards**. Il peut également être utilisé comme réceptacle de métriques pour une architecture Prometheus existante grâce à l'API Remote Write.

NOTRE POINT DE VUE

Essayez cet outil si les contraintes de Prometheus vous découragent ou pour collecter des métriques dans une infrastructure de type Hub & Spoke. Soyez vigilant à l'utilisation de votre backend de stockage et à la consommation CPU/RAM de Mimir en cas de gros volume de données. Mettre en place un throttling sur les requêtes entrantes pourra aider.

ASSESS
2/6 blips

28 Crossplane

Crossplane est un outil d'infrastructure-as-code basé sur Kubernetes. Il permet de créer des ressources Cloud en utilisant des manifestes Kubernetes.

Crossplane est une technologie d'IaC développée par Upbound. **Elle permet de déployer des ressources d'infrastructure en utilisant Kubernetes comme gestionnaire d'état**. Son principe de fonctionnement est similaire à Config Connector de GCP ou AWS Controllers for Kubernetes.

Crossplane se déploie comme un opérateur dans Kubernetes. Pour l'utiliser afin de gérer son infrastructure, il faut déployer des providers qui packagent des CRDs représentant chaque ressource Cloud (exemple pour AWS : une instance EC2, un VPC, une Lambda…). La plupart des providers sont construits à partir de leurs analogues en Terraform.

Associé à des technologies de GitOps telles qu'ArgoCD, **Crossplane peut se transformer en véritable plateforme de self-service Cloud**. L'utilisation de YAML et d'attributs Kubernetes pour définir et relier toute son infrastructure est très intuitive.

La connaissance minimale nécessaire pour utiliser Crossplane est nettement inférieure à celle requise pour Terraform.

En outre, les compositions dans Crossplane (qui sont l'analogue des modules dans Terraform) permettent de mettre à disposition des équipes internes des CRDs personnalisées leur permettant de déployer facilement des parties complexes d'infrastructure (comme un bucket avec toute l'IAM associée).

L'outil s'est considérablement amélioré l'année dernière pour

corriger certains de ses défauts. Les providers ont par exemple été découpés pour réduire la consommation des contrôleurs associés et limiter le nombre excessif de CRDs déployées. Les grosses fonctionnalités manquantes comme le dry-run et l'import de ressources ont aussi été ajoutées.

Cependant, Crossplane garde quelques faiblesses :

- Modifier un champ immutable d'une ressource n'engendre pas son remplacement, il faut gérer manuellement ce cas de figure.
- La dépendance à un cluster Kubernetes pour gérer son infrastructure implique une gestion impeccable de ce dernier.
- Les compositions sont très complexes à gérer pour les non-initiés, mais indispensable pour maintenir un niveau de sécurité élevé sur son infrastructure dans un contexte multi-tenant.
- De nombreux providers ne sont pas encore matures, ou sont en retard sur leur analogue côté Terraform.

NOTRE POINT DE VUE

Nous utilisons aujourd'hui Crossplane pour développer des plateformes self-service pour les développeurs, qui leur permettent de déployer de la même façon des ressources dans Kubernetes et dans le Cloud. L'outil continue à grandir et séduit de plus en plus, si bien qu'il pourrait rapidement devenir un concurrent sérieux des autres technologies d'IaC.

ASSESS
3/6 blips

29 Guacamole

Guacamole scanne votre code Terraform ou Terragrunt pour vous aider à détecter et corriger les mauvaises pratiques et les antipatterns, à la manière d'un linter.

Guacamole est la réponse au constat qu'il n'y a tout simplement pas d'outils de qualité de code relatif à l'IaC.

Beaucoup d'outils existent pour garantir sa sécurité (checkov, tfsec,...), tester le bon fonctionnement (terratest) et même tester le bon formatage du code (terraform fmt).

Mais **aucun outil ne vérifie la sanité du code, c'est à dire s'il est compréhensible**.

C'est dans cette optique que nous avons développé Guacamole.

Nous avons défini des bonnes pratiques d'IaC sur lesquelles nous nous sommes appuyés. Grâce à celles-ci, nous avons créé un outil capable d'effectuer une série de vérifications sur votre codebase :

- La structure d'un répertoire de code IaC
- Le nommage de ressources ou de variables,
- La configuration du provider Terraform
- La configuration des modules et de leurs versions
- La répétition de code.

Cet outil est toujours en cours de développement, mais nous l'utilisons sur nos codebases dans des pipelines de CI. Il nous permet de gagner du temps lors des code reviews ou tout simplement au quotidien en développant avec l'outil installé en pre-commit.

NOTRE POINT DE VUE

Nous le plaçons en assess, car les vérifications sont très opinionnées et auront du mal à s'inscrire dans des codebases existantes. Si vous commencez aujourd'hui un projet avec de l'IaC, nous recommandons de l'utiliser. Vous pouvez whitelister les vérifications non pertinentes pour vous. N'hésitez pas à remonter un bug ou une question en créant une issue GitHub*.

*https://github.com/padok-team/guacamole

ASSESS
4/6 blips

30 OpenTofu

OpenTofu est un fork de Terraform qui ajoute plusieurs fonctionnalités intéressantes, mais ne présente pas de fonctionnalité game changer pour justifier une migration.

#IaC

OpenTofu est un fork open source de Terraform lancé suite au changement de sa licence MPL (Mozilla Public License) vers BUSL (Business Source License) par Hashicorp. Ce changement a pour effet de restreindre les conditions d'utilisation de Terraform. Pour certains usages il est necessaire d'acquérir une licence entreprise pour exploiter Terraform. Notamment dans le cas des outils qui utilisent Terraform et le revendent avec des fonctionnalités supplémentaires.

Même si ce changement de licence n'affecte pas tous les clients, cela est néanmoins un changement de paradigme important pour la suite de Terraform, pour lequel Hashicorp encourage l'utilisation de son outil payant Terraform cloud.

OpenTofu amène des fonctionnalités encore jamais implémentées par Terraform, mais suit aussi les nouvelles apportées par Hashicorp en les reproduisant sans recopier le code (pour éviter les soucis de plagiat).

Les nouvelles fonctionnalités d'OpenTofu sont notamment :

- Le chiffrement end-to-end du state
- l'option -concise qui permet de retirer tout le bruit pour un apply
- L'utilisation de variables et locals est autorisée sur les modules sources et les configurations de backend.

OpenTofu montre une promesse intéressante pour l'avenir, spécifiquement grâce à sa communauté active et à son développement open source.

Cependant, pour les utilisateurs déjà bien ancrés dans l'écosystème Terraform, il manque peut-être une innovation majeure qui les inciterait à abandonner un outil éprouvé au profit d'un autre en plein développement.

NOTRE POINT DE VUE

En résumé, OpenTofu représente une alternative solide avec des fonctionnalités enrichies, mais sans offre révolutionnaire. Elle reste un choix à considérer attentivement en fonction des besoins spécifiques et des contraintes de chaque organisation.

ASSESS
5/6 blips

31 Pulumi

Pulumi est un outil d'Infrastructure as Code qui offre de nombreuses possibilités, mais n'est pas encore le premier choix pour construire son infrastructure en IaC.

#IaC

Terraform est peut-être le leader sur l'IaC mais Pulumi est un concurrent sérieux avec **une approche utilisant des langages comme Python ou GO**. Cela permet d'**écrire du code conditionnel plus facilement, chose complexe en Terraform**.

Pulumi propose 2 fonctionnalités qui se différencient des autres outils IaC :

- **Les providers natifs** qui sont automatiquement mis à jour en fonction de l'API officielle des Cloud Providers.
- **La gestion de secrets par chiffrement**, qui permet d'écrire des données sensibles directement dans le code. En effet, elles sont chiffrées avec des clés venant de providers comme AWS KMS, Hashi Vault ou encore Cloud Kms et déchiffrées au run time juste-à-temps.

Utiliser Pulumi est un bon compromis **pour les équipes composées uniquement de développeurs** et qui souhaitent rester sur un langage familier. Cela est avantageux, car les process de maintenance et les bonnes pratiques en place permettent de garantir une qualité de code importante.

Pour autant Pulumi vient avec des limitations.

- En fonction du langage, la gestion de dépendances avec par exemple les "node_modules" peut devenir volumineuse avec le scaling de la code base.
- La qualite du code est difficile à tracker. En effet, il n'y a pas de SAST (Checkov ou tfsec) pour Pulumi.

NOTRE POINT DE VUE

Si vous avez un **besoin précis d'utiliser des langages** comme le GO ou le Python dans votre stack, démarrer avec Pulumi sera plus simple. Cependant, Pulumi ne résout pas les défis fondamentaux de Terraform concernant la création, l'organisation et la maintenance du code IaC. Si vous utilisez déjà Terraform pour gérer votre infrastructure, ne migrez pas vers Pulumi.

ASSESS
6/6 blips

32 Terratest

L'infrastructure étant la base de toute application robuste, elle doit être testée ! Terratest répond justement à ce problème en étant l'une des rares librairies de test pour Terraform.

#IaC

Terratest est la **référence pour tester son code Terraform**. Codée en Go, cette librairie permet d'écrire des tests unitaires pour Terraform et Terragrunt.

En effet, **Terratest permet de déployer une infrastructure et de réaliser des tests sur** :

- Des appels HTTPS pour voir si des load balancers sont bien fonctionnels
- Des requêtes API pour vérifier la réponse de l'API gateway
- Des connexions SSH vers des serveurs bastion
- Le bon statut d'une ressource
- L'application d'un Terraform Apply sans erreur

Ce sont des tests qui deviennent essentiels pour des infrastructures grandissantes car des erreurs peuvent rapidement apparaître, dues à des changements de version de Terraform ou du provider, et surtout pour garantir la non-régression des fonctionnalités existantes.

Cependant, il existe **des freins à son utilisation par les modules open sources maintenus par les Cloud Providers** :

- La mécanique de conception du test pour valider le fonctionnement de l'infrastructure n'est pas un geste facile et documenté
- La disponibilité d'un environnement pour réaliser ces tests engendre des coûts supplémentaires, même si c'est sur une courte période

NOTRE POINT DE VUE

On le positionne en "Assess" car aujourd'hui ce n'est pas encore un standard de l'industrie.
En note, nous l'avons également utilisé pour tester nos packages Helm et nous en sommes satisfaits !

LE CADRAN

Secure

17 BLIPS | 8 ADOPT | 5 TRIAL | 3 ASSESS | 1 HOLD

Pour ce cadran "Secure", nous avons choisi parmi un large panel d'outils open source ou de patterns de sécurité particulièrement adaptés à des environnements cloud-native.

Nous avons volontairement passé sous silence tous les outils en SaaS extrêmement coûteux qui inondent aujourd'hui le monde de la sécurité Cloud. Nous croyons qu'il est tout à fait possible d'atteindre un haut niveau de sécurité simplement grâce à des bons designs d'architecture, une CI/CD bien outillée et des outils open source.

Les blips présentés dans cette section couvrent un large spectre de sécurité opérationnelle : sécurité réseau, gestion des accès, gestion des secrets, sécurité Kubernetes, sécurité des conteneurs, outillage CI/CD.

PAR **THIBAULT LENGAGNE**
Head of Security

ADOPT
1/8 blips

33 Checkov

Checkov est un outil d'analyse de code statique (SAST) pour votre Infrastructure as Code, permettant de vérifier et de renforcer l'application des bonnes pratiques, notamment sur la sécurité.

#ShiftLeftSecurity #PoliciesAsCode

Checkov, et les SAST de manière générale, s'inscrivent dans le paradigme récent de la sécurité Shift Left. **Il s'agit d'une philosophie qui vient inscrire la sécurité plus tôt dans le cycle de développement d'une application**. Là où le paradigme traditionnel voit la sécurité être implémentée à la suite du développement du logiciel, au moment de la phase de test, la sécurité Shift Left voit cet intérêt pour la sécurité être déplacé au moment du design et du développement de l'application.

L'outil supporte l'analyse des configurations de plusieurs technologies comme : Terraform, Helm, Docker, Ansible et la plupart des gestionnaires de CI/CD. Il est également capable de détecter la présence de secrets en clair dans le code.

Nous avions déjà placé Checkov en "Adopt" sur notre dernier Tech Radar, avant la sortie en octobre 2023 de la version 3.0. Cette nouvelle version implémente la compatibilité avec Jenkins et Bicep, mais également la possibilité de créer ses politiques custom. La création de politiques personnalisables est simple et peut se faire en python ou même avec du YAML. Nous avons trouvé cette fonctionnalité utile pour vérifier l'application de tags spécifiques par exemple, ou d'autres bonnes pratiques qui ne se réfèrent à aucun framework de sécurité.

S'adaptant aux tendances, **Checkov nous offre aussi désormais la possibilité de s'interfacer avec OpenAI** pour obtenir des analyses de résultats plus détaillées et guidées.

NOTRE POINT DE VUE

Nous continuons d'utiliser Checkov au quotidien dans nos process de développement mais aussi lors de nos différentes missions d'audit de sécurité infrastructure. C'est un très bon outil d'audit de contrôle dans un contexte réglementaire comme ISO 27001, PCIDSS ou encore SOC2.

ADOPT
2/8 blips

34 External Secrets

External Secrets permet d'utiliser des systèmes de gestion de secrets externes pour ajouter des secrets en toute sécurité dans Kubernetes.

#GestionDesSecrets

Stocker ses secrets de manière centralisée est un des grands principes de la sécurité. External Secrets a été créé par l'engineering team de GoDaddy en 2019 suite à l'expression de ce besoin : **mettre à disposition des secrets stockés dans un système central à des applications Kubernetes**.

External Secrets propose de nombreuses intégrations avec les systèmes de gestion de secrets classiques comme Azure, AWS, GCP, Vault, Gitlab, Oracle ou IBM. Notons également le puissant moteur de templating permettant de créer des secrets formatés comme bon vous semble. Par exemple, vous pouvez créer un secret Kubernetes TLS à partir d'une archive PKCS#12.

En termes de gestion des droits et du respect du moindre privilège, **External Secrets permet de restreindre les droits par namespace via l'utilisation de SecretStore**. Si vous souhaitez un contrôle plus fin sur les droits, par exemple limiter l'accès à certains prefix de path, il faudra utiliser des Admission Webhook comme Kyverno.

Si le secret d'origine est modifié, External Secrets va mettre à jour la valeur du secret Kubernetes. L'intervalle de temps de polling est paramétrable par ressource ExternalSecret. Attention, la mise à jour d'un secret Kubernetes n'entraînera pas le rollout d'un pod qui utiliserait ce secret. Si vous souhaitez ce genre de fonctionnalité, il faudra utiliser un outil plus puissant comme Vault Secret Operator.

NOTRE POINT DE VUE

Il existe évidemment d'autres outils remplissant la même mission et apportant différentes fonctionnalités : Vault Operator, KSOPS, Sealed Secrets, ArgoCD Vault plugin… N'hésitez pas à comparer ces solutions et choisir celle qui vous convient.

ADOPT
3/8 blips

35 Helm Secrets

Helm Secrets sécurise les secrets stockés sur Git via chiffrement, simplifiant leur gestion pour les équipes qui débutent.

#GestionDesSecrets

On vous l'a maintes fois répété : **stocker des secrets en clair dans un dépôt Git constitue une faille de sécurité majeure**. Les informations sensibles comme les clés d'API, les mots de passe et les certificats peuvent être facilement exposées, ouvrant la porte à des cyberattaques. Il est par conséquent crucial de chiffrer vos secrets avant de les ajouter à un dépôt Git pour garantir qu'ils restent bien… secrets.

Dans ce contexte, **Helm Secrets est un plugin Helm qui simplifie la gestion des secrets dans un dépôt Git**, permettant de les stocker de manière sécurisée tout en les rendant accessibles pour les déploiements Kubernetes via ArgoCD par exemple. Helm Secrets utilise des outils de chiffrement comme AWS KMS ou GCP KMS couplés à l'outil SOPS pour **chiffrer les fichiers secrets localement** puis les déposer sur le dépôt Git. Cette approche est particulièrement pratique pour une équipe de développement peu expérimentée et encore un peu éloignée des pratiques cloud-native, car elle réduit l'apprentissage nécessaire.

Point intéressant, **les fichiers YAML chiffrés par Helm Secrets ne sont pas complètement opaques** : les équipes peuvent voir les clés YAML présentes à l'intérieur sans compromettre la sécurité des valeurs, ce qui facilite les revues de code.

En termes de mise en place, Helm Secrets nécessite simplement l'installation du plugin dans la stack de CI/CD en charge de déployer les charts Helm. Niveau sécurité, comme les secrets restent chiffrés au repos et en transit, cela coche pas mal de cases sans complexité excessive.

NOTRE POINT DE VUE

Comparé à d'autres outils comme External Secrets ou Sealed Secrets qui demandent de rédiger des templates compliqués, **Helm Secrets se distingue par sa simplicité**, ce qui en fait un choix judicieux pour des équipes recherchant un équilibre entre sécurité et facilité d'utilisation.

ADOPT
4/8 blips

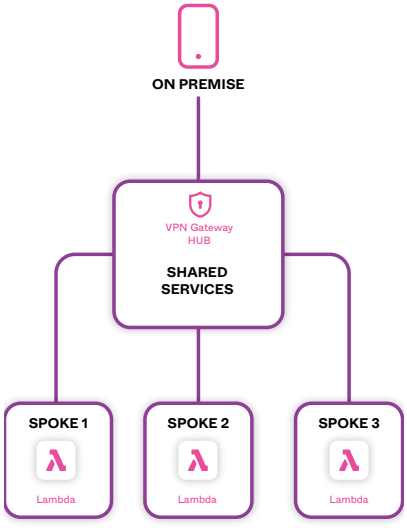
36 Hub and Spoke

Le Hub and Spoke isole les ressources des applis ou des services (Spoke) avec un compte central (Hub) qui gère les communications réseau et partage les outils entre les spokes.

#Networking

Le modèle Hub and Spoke est utilisé pour gérer les comptes cloud. Si la création de ressources dans le cloud est simple, les organiser à grande échelle et garantir leur sécurité est bien plus complexe. Ce modèle est principalement adopté par les entreprises ayant de nombreuses applications et un besoin élevé de gouvernance.

Le compte Hub est le compte central. Il **permet de centraliser les communications** (entrantes, sortantes), de les monitorer et de renforcer l'isolation entre les réseaux des différents environnements ou des différentes applications. Il **permet aussi de centraliser les politiques de sécurité** qui seront ensuite héritées par les comptes Spokes.



Les Spokes contiennent tout ce qu'il faut pour héberger des workloads. Ces comptes sont provisionnés en tenant compte des besoins de vos applications, **en héritant des bonnes pratiques et politiques de votre organisation** et en étant connectés au Hub pour profiter de la gestion centralisée du réseau. Le niveau d'autonomie des équipes sur les Hub sera à définir en fonction de votre organisation et de vos contraintes de sécurité.

NOTRE POINT DE VUE

Une infrastructure **Hub and Spoke permet de garantir un haut niveau de sécurité sur une infrastructure à grande échelle.** Implémenter un Hub and Spoke est souvent utile pour des grandes organisations, mais il a tout intérêt à être implémenté dans des petites organisations pour avoir des fondations solides et scalables.

ADOPT
5/8 blips

37 Kyverno

Kyverno est une solution de Policy as Code open source, permettant d'assurer le respect de bonnes pratiques de sécurité lors du déploiement dans Kubernetes.

#PoliciesAsCode

La plupart des paramètres de sécurité des applications Kubernetes doit être vérifiée au plus tôt, notamment dans la CI/CD. Mais **la validation de politiques au déploiement, directement intégrée au moteur de validation de Kubernetes, permet d'assurer le respect des bonnes pratiques**.
De plus, elle limite les actions d'un attaquant en cas de compromission.

Kyverno permet d'étendre les possibilités de validation de sécurité natives de Kubernetes (notamment les Pod Admission Policies, introduits avec Kubernetes 1.25 ou encore le RBAC) et peut **s'y substituer ou s'appliquer de manière complémentaire**.

Intégré depuis 2020 au sein de l'incubateur CNCF, et en statut Incubating depuis 2022, **Kyverno est aujourd'hui une valeur sûre dans l'écosystème sécurité de Kubernetes**.

La puissance de Kyverno réside dans la **simplicité d'écriture des politiques en YAML et dans la variété de types de politiques proposés** : validation, mutation ou génération de ressources à la volée.

Si de nombreux outils permettent aujourd'hui d'auditer et de valider la configuration des applications dans Kubernetes, Kyverno est une des rares solutions proposant une gestion avancée des exceptions, répondant ainsi aux contraintes opérationnelles d'un cluster de production.

Les politiques de génération de ressources sont particulièrement intéressantes, y compris dans des environnements gérés en GitOps, pour améliorer l'expérience utilisateur dans la gestion des NetworkPolicies (les règles de pare-feu dans Kubernetes), en permettant par exemple :

- De descendre automatiquement des NetworkPolicies bloquant les accès aux endpoints de métadonnées des Cloud Providers à la création d'un namespace ;
- D'autoriser les ingress controllers à contacter les services exposés sans avoir à explicitement créer une NetworkPolicy, celle-ci étant automatiquement créée par Kyverno.

Grâce aux **rapports générés par Kyverno**, il est également possible d'obtenir des métriques sur la conformité du cluster vis-à-vis des politiques configurées.

NOTRE POINT DE VUE

Kyverno est donc bien plus qu'un outil d'audit de configuration et permet aux équipes de définir une politique de sécurité et de s'assurer de son respect dans les environnements Kubernetes.

ADOPT
6/8 blips

38 NetworkPolicies

Les NetworkPolicies sont essentielles à la sécurité d'un cluster Kubernetes. Par défaut, le trafic réseau n'y est pas restreint, et les NetworkPolicies résolvent ce problème.

#Networking

Le filtrage réseau est l'une des meilleures mesures pour limiter l'impact d'une compromission dans un système d'information, et il en va de même dans un cluster Kubernetes. Cependant, **le filtrage réseau est bien trop souvent négligé dans un cluster (il n'y en a pas par défaut !)**, là où il est vu comme essentiel dans un réseau classique.

Les NetworkPolicies offrent une solution à ce problème en permettant de définir des règles de filtrage réseau entre les pods.

La mise en œuvre technique des NetworkPolicies dépend de la CNI installée et peut se faire sous différentes formes : création de règles iptables, module eBPF, etc. Le comportement par défaut (autoriser le trafic ou bloquer le trafic) va également dépendre de cette dernière. Il est important de bien se renseigner sur l'implémentation des NetworkPolicies par la CNI choisie avant d'en mettre en place.

Les ressources NetworkPolicies font partie de l'API native Kubernetes. Toutefois, pour résoudre les limitations de la ressource native, certaines CNI (telles que Calico ou Cilium) permettent, en complément, d'utiliser des ressources spécifiques (CRD Kubernetes) pour définir les règles de filtrage.

NOTRE POINT DE VUE

Le filtrage réseau précis peut être complexe, mais **les NetworkPolicies sont cruciales pour sécuriser un cluster Kubernetes.** Nous recommandons de commencer par des politiques simples qui amélioreront votre sécurité, comme isoler les composants de tooling des applications ou bloquer l'accès aux métadonnées des nœuds dans un environnement managé.

ADOPT
7/8 blips

39 SonarQube

SonarQube est un outil open source d'analyse permettant d'assurer la qualité et la sécurité de votre code. Il supporte un grand nombre de langages de programmation différents.

#ShiftLeftSecurity

La qualité et la sécurité du code applicatif doivent être assurées tout au long du processus de développement, notamment par des outils intégrés à la CI/CD. **SonarQube combine une analyse statique et dynamique du code** à la recherche de bugs, de code smell et de vulnérabilités. Cela permet d'assurer une bonne sécurité, fiabilité et maintenabilité du code.

SonarQube possède aussi un **grand nombre de jeux de règles disponibles** (e.g. Top 10 de l'OWASP) **sur plus d'une vingtaine de langages de programmation**. De plus, vous pouvez ajouter vos propres règles.

Un niveau de criticité est également proposé pour chaque résultat, et permet d'avoir un score global d'une analyse divisée en 4 catégories. Il est ainsi possible de définir des standards bloquants sur les pipelines de déploiement, et d'imposer un certain niveau de qualité ou de sécurité du code tout au long du processus de développement.

Une console d'administration intuitive est disponible, où sont recensés les différents problèmes rencontrés dans le code. Elle permet d'avoir une vue d'ensemble sur les différents projets scannés.

Attention néanmoins, la console d'administration contient un grand nombre d'informations sensibles et son accès doit être correctement protégé. Le système de gestion des droits RBAC dans SonarQube est simple, mais efficace.

SonarQube sera un allié précieux dans une stratégie de sécurité shift left, permettant de détecter et de corriger les problèmes de sécurité dès les premières étapes du développement. Une approche proactive de la sécurité plutôt que réactive est alors favorisée.

À PROPOS DU PRIX

SonarQube est disponible en SaaS (SonarCloud) ou en Open Source. Nous recommandons de commencer par le SaaS qui reste abordable au départ : 10€ par mois en dessous de 100k lignes de code.

ADOPT
8/8 blips

40 Vault

Vault est un outil de gestion de secrets incontournable pour toutes les organisations.

#GestionDesSecrets

Hashicorp Vault est l'**outil de gestion de secrets le plus plébiscité par la communauté**. Cela est possible grâce aux innombrables fonctionnalités et améliorations que Vault a apportées à ce secteur :

- La génération de secrets dynamiques.
- La gestion de certificats directement via une API.
- La gestion fine d'accès aux secrets en suivant un pattern RBAC (Role-Based Access Control) applicable à des utilisateurs, groupes ou machines.
- L'intégration de différentes sources d'authentification (e.g Kubernetes).

Deux différences majeures font que Vault s'impose face à ses concurrents. Tout d'abord, son **intégration avec différentes sources d'identités** (e.g AWS, GCP, Kubernetes) qui permettent à vos workloads de s'authentifier et d'être autorisés à récupérer automatiquement leurs différents secrets. Mais également les **différents moteurs de secrets dynamiques** (Database, AWS, GCP) **qui permettent une rotation des secrets à chaque utilisation.**

Nous n'avons qu'une seule mise en garde quant à son adoption en mode auto-hébergé : **Vault est un outil complexe** et peut ne pas convenir à une petite organisation. Il ajoute comme toute technologie un coût d'opérabilité, et une mauvaise gestion de vos backups ou une mauvaise anticipation d'une mise à jour pourrait vous mener droit dans le mur.

Si votre équipe est déjà surchargée, préférez l'utilisation de la version SaaS de Vault ou alors les services de gestion de secrets de vos Cloud Providers.

NOTRE POINT DE VUE

Nous ne pouvons que le recommander aujourd'hui tant il nous a permis de répondre à des problématiques complexes sans ajouter un coût de maintenance prohibitif.

TRIAL
1/5 blips

41 Architecture
Zero-Trust

Zero-Trust est un modèle basé sur l'absence de confiance entre les parties : toutes les communications doivent être autorisées.

#ZeroTrust

Le principe de base de l'architecture Zero-Trust est de ne faire confiance à personne ou à aucun périphérique par défaut, même s'ils se trouvent à l'intérieur du réseau. Il est aussi nécessaire de **vérifier en permanence l'identité et les autorisations** de tout ce qui tente d'accéder aux ressources de l'entreprise. Aucune confiance implicite n'est accordée à un utilisateur ou à un service en fonction de l'endroit d'où provient la requête.

Attention, Zero-Trust ne signifie pas qu'il faut arrêter de restreindre l'accès aux ressources via du filtrage réseau ! Le modèle Zero-Trust part simplement du principe que le réseau sera nécessairement compromis, ou que le périmètre sera défaillant. Cela contraint donc les utilisateurs et les appareils à prouver leur légitimité.

Le Zero-Trust est particulièrement adapté aux infrastructures Cloud, qui possèdent souvent une typologie réseau complexe, composées de plusieurs réseaux virtuels interconnectés, potentiellement exposés sur internet. Dans ce contexte, la notion même de périmètre finit par perdre son sens, notamment dans les infrastructures serverless… C'est pour cela que l'IAM des Cloud Providers est fortement inspirée du modèle Zero-Trust et demande à chaque utilisateur ou service de prouver son identité avant de pouvoir accéder à des ressources.

Cette approche est séduisante, mais **complexe à mettre en place**. Chaque utilisateur et chaque application doit utiliser un système d'authentification centralisé (par exemple le MFA pour les utilisateurs ou les certificats TLS pour les serveurs). Généralement, un gestionnaire d'identité (Cloudflare Zero-Trust, Tailscale) fait le lien entre les terminaux externes (postes utilisateurs) et les services hébergés dans l'infrastructure.

NOTRE POINT DE VUE

Le concept de Zero-Trust est donc particulièrement adapté pour les entreprises ayant un **besoin accru de protection de leurs données,** mais nécessite un gros investissement de mise en place.

TRIAL
2/5 blips

42 BuildKit

Buildkit est un moteur de build d'images de conteneurs avec un faible niveau de privilèges tout en prenant en charge toutes les instructions possibles d'un Dockerfile.

Le build d'images de conteneurs est aujourd'hui une problématique centrale de la CI/CD des applications conteneurisées. Mais c'est aussi souvent une opération coûteuse qui peut impliquer la compilation de binaires, et donc une forte charge RAM et CPU.

Alors, afin de s'adapter à une charge forte, mais très variable en fonction des développements en cours, il est confortable de réaliser ces builds dans des clusters Kubernetes habituellement partagés avec d'autres outils.

Pour assurer la sécurité de ces clusters, il est important que les outils qu'ils hébergent, et donc notamment ceux de build d'images de conteneurs, soient exécutés sans privilèges.

Or, historiquement, les **builds d'images de conteneurs nécessitent un processus privilégié**. Par conséquent, un développeur qui a le droit de lancer une pipeline de build pourrait prendre le contrôle de tous les conteneurs qui tournent sur le même nœud.

Heureusement, **de nombreuses évolutions du noyau Linux réalisées ces dernières années ont permis de voir apparaître la possibilité de lancer des conteneurs quasiment sans privilèges**. Et les moteurs de build se sont adaptés en conséquence. Ainsi des outils comme Buildkit, Buildah, ou encore Kaniko, prennent désormais en compte chacun à leur manière ces nouvelles possibilités.

NOTRE POINT DE VUE

Nous favorisons **Buildkit** sur nos projets car il semble être le seul à **vraiment gérer toutes les instructions possibles d'un Dockerfile**. De plus, il est bâti sur un modèle client/serveur qui permet de profiter d'un cache commun et d'offrir de belles performances, donc de diminuer le temps nécessaire au build. Un élément de sécurité qui accélère les développeurs, c'est possible !

TRIAL
3/5 blips

43 Cilium

Cilium est un plugin réseau pour Kubernetes qui offre des fonctionnalités avancées de sécurité et d'observabilité.

#Networking

Cilium est un **projet open source de CNI (Container Network Interface) pour Kubernetes**, gradué CNCF en octobre 2022. Ce plugin réseau gère le trafic entre les charges de travail d'un cluster et offre des fonctionnalités avancées telles que le chiffrement transparent du trafic entre pods (sans sidecar proxies), un filtrage réseau avancé avec les NetworkPolicies, une observabilité accrue du trafic, ou la communication multi-cluster.

La force de Cilium est que **ce CNI est basé eBPF** (Extended Berkeley Packet Filter), une technologie du noyau Linux permettant d'exécuter des programmes isolés dans l'espace noyau. **eBPF étend de manière sûre et efficace les capacités du noyau** sans modifier son code ou ajouter des modules, **offrant des performances supérieures à iptables** utilisé par kube-proxy. Cilium n'a pas besoin de module noyau supplémentaire et fonctionne sur tout serveur avec un noyau Linux récent (>= 4.19).

L'outil offre des fonctionnalités essentielles pour un cluster Kubernetes central et critique pour votre infrastructure, notamment en sécurité et observabilité. Par exemple, vous pouvez effectuer un filtrage réseau couche 7 avec la CRD CiliumNetworkPolicy ou obtenir une visibilité approfondie sur le trafic de couche 7 grâce aux métriques de Cilium.

Cilium **permet aussi de chiffrer facilement le trafic entre les pods.** Pour ceux utilisant un Service Mesh pour le chiffrement mTLS, Cilium apporte une solution plus simple.

NOTRE POINT DE VUE

Cilium est devenu le CNI de choix pour Kubernetes, utilisé par Datadog et dans les dataplane v2 de GKE. Mais il est complexe à utiliser, requiert des compétences techniques avancées, et n'est pas toujours configurable dans les clusters managés (par exemple, GKE ne permet pas de configurer directement le chiffrement entre pods via le CNI). De plus, beaucoup de fonctionnalités sont encore en bêta.

TRIAL
4/5 blips

44 Falco Security

Falco est un outil de détection d'intrusion pour Kubernetes. Il repose sur l'analyse dynamique des appels systèmes et des audits logs pour lever des alertes en cas de comportement suspect.

Falco est un outil open source développé par Sysdig qui fait partie de la CNCF depuis 2018. **Il permet de détecter les comportements suspects dans les environnements conteneurisés**. Il s'intègre donc parfaitement à Kubernetes.

Falco fonctionne en ingérant les différents appels systèmes effectués sur les machines hôtes ainsi que les événements de l'API Server. Cela représente un avantage notable par rapport aux autres HIDS (Host-based Intrusion Detection Systems). En corrélant ces logs avec des règles configurables, Falco est capable de détecter les menaces potentielles et d'envoyer des alertes fournissant des informations détaillées sur les événements déclencheurs. Il peut notamment identifier les mutations de binaires, la lecture des secrets, etc. **Falco est hautement personnalisable**, permettant aux utilisateurs de créer leurs propres règles et alertes pour répondre à leurs besoins de sécurité spécifiques. Il est possible de gérer ces règles as code grâce au GitOps.

Falco est un élément essentiel pour assurer la sécurité d'un cluster Kubernetes. Nous recommandons son utilisation dans la mesure où vos équipes ont le temps de traiter et de configurer les alertes. La grande difficulté dans sa mise en place réside dans le fine-tuning de la configuration pour produire le bon volume d'alerte. Il convient de diminuer au maximum le nombre de faux positifs avec un système de whitelisting.

Il est à noter également que des problèmes persistent quant à l'interconnexion de Falco avec le noyau : par exemple, dans un cluster EKS, les modules noyau Falco sont publiés en général 2 semaines après la publication d'une nouvelle AMI par AWS, ce qui retarde la mise à jour des nœuds.

NOTRE POINT DE VUE

Nous recommandons donc cet outil dans la mesure où votre équipe possède la bande passante nécessaire pour traiter les alertes au quotidien, autrement vous alimenterez juste la charge des personnes responsables de la réponse à incident.

TRIAL
5/5 blips

45 Linkerd

Linkerd permet de mettre en place simplement le chiffrement et l'autorisation des communications dans les clusters Kubernetes.

#Networking

Linkerd est un Service Mesh pour Kubernetes, gradué par la CNCF. **Il offre des fonctionnalités avancées pour les communications au sein du cluster**, telles que le chiffrement inter-Pods et des politiques de Zero-Trust pour une sécurité accrue. Il permet également une observabilité poussée des communications inter-Pods, ainsi que l'injection de fautes.

Linkerd propose une architecture très simple comparé à d'autres Services Mesh comme Istio. Cela est notamment **dû au nombre restreint de fonctionnalités de Linkerd**, qui peut s'étendre grâce à l'utilisation de plugins : par exemple le tracing grâce à Jaeger ou les canary releases grâce à Flagger. Certaines fonctionnalités proposées par ses concurrents sont toujours absentes dans l'outil, notamment le JWT header based routing. Cette architecture simplifiée réduit le coût de mise en place de ce service Mesh, qui ne nécessite pas son propre Ingress controller. Cela rend son intégration très facile même sur des clusters déjà outillés. De plus, il peut s'interfacer avec des outils comme Vault ou cert-manager pour gérer le renouvellement automatique des certificats mTLS utilisés. Linkerd se démarque de ses concurrents grâce à **d'excellentes performances** dûes à son proxy minimaliste écrit en Rust qui réduit la charge pour le cluster.

Le principal point faible de Linkerd est qu'il n'y a plus de version stable Open Source depuis la 1.24, mais seulement des versions "edge" qui peuvent amener à des breaking changes d'une version à l'autre. Pour garder des versions stables, il faudra passer à la version enterprise (Buoyant).

NOTRE POINT DE VUE

Nous recommandons Linkerd, qui a l'avantage de réduire l'impact opérationnel sur le cluster par rapport à ses concurrents. Il offre la plupart des fonctionnalités attendues, notamment via son système de plugin. Attention à bien tester les nouvelles versions sur des clusters de non-production avant passage en production puisqu'il n'y a plus de support stable.

ASSESS
1/3 blips

46 Boundary

Boundary est un outil voué à remplacer les systèmes existants qui permettent d'accéder à vos réseaux internes.

#Bastion

Boundary est un outil développé par Hashicorp et open sourcé en 2020. Il a pour vocation de **remplacer vos solutions de Bastion, voire de VPN**, en améliorant leur gestion, leur sécurité ainsi que leur expérience utilisateur.

Boundary permet d'**obtenir la même expérience utilisateur que la fonctionnalité de "port-forwarding" de Kubernetes mais sur l'ensemble de votre infrastructure** (e.g. BDD, VMs, WebServices etc.). Il permet l'**auditabilité et la ségrégation** d'accès via des audits logs, la possibilité de révoquer des sessions en cours et également toute une logique RBAC (Role-Based Access Control) de gestion d'accès.

Il se décompose en :

- Un contrôleur central qui contient la configuration des différentes cibles auxquelles vos employés ont besoin d'accéder, ainsi que la gestion des droits.
- Des workers qui ont accès à vos différents réseaux internes et qui servent de passerelle.

La grande force de Boundary réside dans sa **simplicité d'installation et d'opérabilité**. Sa configuration peut être complexe, mais son intégration avec Terraform lui permet d'être automatisée de la même façon que le reste de la configuration de votre infrastructure. Nous apprécions également son intégration avec Vault, qui permet aux utilisateurs d'accéder à des ressources sans avoir à connaître aucun secret.

NOTRE POINT DE VUE

Boundary est encore jeune et nous ne sommes pas encore pleinement satisfaits de l'expérience utilisateur qu'il propose, notamment lors de l'ouverture de session vers un cluster Kubernetes. Nous sommes tout de même confiants pour la suite et pensons que le projet deviendra une référence dans ce type d'outils.

ASSESS
2/3 blips

47 Istio

Istio est un outil complexe permettant un contrôle extrêmement fin des différentes communications entre vos applications.

#Networking

Istio est un Service Mesh comme Linkerd qui offre de nombreuses fonctionnalités pour la sécurisation et l'observabilité de l'ensemble des communications au sein d'un cluster mais également vers l'extérieur.

Istio est un incontournable qui a fait ses preuves et dont tout le monde connaît l'existence.

En revanche, **l'outil est difficile à mettre en place et à opérer**, ce qui peut bloquer de nombreuses personnes quant à son adoption.

Nous avons déjà essuyé beaucoup de plâtres dans nos projets à cause de lui, notamment le passage à la version 1.6 qui a marqué de nombreux utilisateurs. Les mainteneurs de l'outil ont néanmoins mis l'accent sur cette problématique l'année passée si bien qu'il est devenu nettement plus simple à utiliser. Un Helm Chart est en effet maintenant disponible pour Istio, ce qui permet de le gérer en GitOps au lieu d'utiliser la CLI istioctl.

Cependant, la principale motivation pour l'installation d'Istio est le chiffrement des communications inter-clusters pour répondre aux exigences de sécurité. De fait, nous favorisons l'utilisation d'outils plus simples et dont la charge sur les clusters est moins importante comme Linkerd.

Mais dans un contexte de forte contrainte de sécurité, où notamment le contrôle des flux de sortie est nécessaire, Istio reste un outil qui répondra parfaitement au besoin. Les fonctionnalités qu'il propose sont très matures et généralement plus exhaustives que celles de ses concurrents. L'outil intègre par exemple Kiali qui offre une interface graphique permettant de visualiser l'ensemble des flux au sein d'un cluster.

NOTRE POINT DE VUE

Istio reste un outil complexe dont **l'adoption nécessite une bonne évaluation préalable.** Istio rajoute un coût de maintenance assez important qui pourrait affecter des équipes qui sont parfois déjà surchargées.

ASSESS
3/3 blips

48 OAuth2 Proxy

OAuth2 Proxy est un reverse proxy qui sert à protéger des ressources exposées publiquement par une authentification via OIDC (Google, Keycloak, GitHub, etc.).

#ZeroTrust

OAuth2 Proxy permet de **protéger ses applications avec de l'authentification assurée par un fournisseur d'identité**. Les applications internes ou les assets de staging peuvent, par exemple, être sécurisés de cette manière. OAuth2 Proxy fait office de reverse proxy : ainsi lorsqu'un utilisateur veut se connecter à une application protégée, il doit d'abord passer par OAuth2 Proxy, s'authentifier, et accéder à l'application uniquement si son identité le permet.

OAuth2 Proxy permet avant tout de protéger toutes les applications très simplement avec de l'OIDC, même si elles ne sont pas initialement compatibles. De plus, **il centralise l'authentification avec du SSO** pour les actifs réservés à un usage interne.

Le contrôle d'accès reste cependant assez simple. OAuth2 Proxy accepte ou non les utilisateurs en s'appuyant sur les scopes du token JWT fourni par l'IdP. Il ne permet pas facilement de créer des groupes d'utilisateurs ou de donner des accès spécifiques à certaines applications. De fait, OAuth2 Proxy peut être utilisé comme moyen d'autorisation, mais pas comme moyen de gestion des identités.

De plus, la surcouche d'authentification peut compliquer l'intercommunication des différents composants. Par exemple, un front ne pourra pas communiquer avec un service protégé par OAuth2 Proxy s'ils n'ont pas le même nom de domaine.

NOTRE POINT DE VUE

Nous recommandons donc l'utilisation d'OAuth2 Proxy dans des cas relativement simples, comme la protection d'outils internes. Attention, **OAuth2 Proxy ne remplace pas non plus l'authentification portée par les applications,** qui elle, protège contre d'autres types d'attaques comme les SSRF.

HOLD
1/1 blip

49 Open Policy Agent

Open Policy Agent (OPA) est un outil open-source centralisant et unifiant la gestion des politiques de sécurité et de conformité des infrastructures, notamment Kubernetes.

#PoliciesAsCode

Open Policy Agent est un moteur de Policy as Code qui permet d'appliquer ou de vérifier l'application de bonnes pratiques dans des configurations d'infrastructure et de tooling.

OPA est utilisable pour valider les configurations de multiples outils et implémentations. De notre côté, nous nous sommes beaucoup intéressés à son intégration dans Kubernetes mais l'outil permet aussi de vérifier des configurations Terraform, Docker, Kafka, Envoy et même des configurations SSH.

Le moteur OPA se base sur un langage déclaratif appelé Rego. L'utilisation de ce langage est l'une des raisons pour lesquelles nous avons placé OPA dans la catégorie Hold. En effet, **ce langage est complexe et difficile à prendre en main** et il est dommage qu'aucun autre langage comme YAML voire JSON ne soit compatible avec le moteur.

Open Policy Agent possède un projet d'intégration avec Kubernetes appelé Gatekeeper. **Ce projet permet d'implémenter** (tout comme Kyverno) **des politiques pour vérifier ou appliquer des bonnes pratiques de sécurité dans Kubernetes**. L'utilisation de cet outil est bien plus complexe et lourde que Kyverno. Contrairement à Kyverno qui nous permet de créer des politiques à la demande, OPAGatekeeper impose la mise en place de CRDs afin de créer des politiques qui seront appliquées aux ressources pour lesquelles vérifier la bonne configuration.

NOTRE POINT DE VUE

Open Policy Agent est un outil puissant mais dont **la grande complexité éclipse la qualité** au profit d'autres moteurs de policy-as-code plus spécifiques comme Kyverno pour Kubernetes ou Checkov pour Terraform.

LE CADRAN

Empowering

14 BLIPS | 5 ADOPT | 4 TRIAL | 3 ASSESS | 2 HOLD

Nous définissons le cadran “Empowering” comme l’ensemble des techniques et technologies qui permettent d’offrir une expérience fluide aux développeurs utilisant nos infrastructures.

Les technologies que nous avons étudiées se placent sur la frise chronologique suivie par votre projet. L'écriture de la première ligne de code est facilitée par les environnements de développement distants ou "à la volée". Les outils de CI/CD, établis (Jenkins, CI/CD intégrées) comme visionnaires (ArgoCD, Argo Rollout) vous aideront à continuellement tester et déployer vos nouvelles fonctionnalités. Enfin, les bugs seront identifiés et éliminés au moyen d'outils de logging et de tracing performants (Sentry, Loki).

Instrumenter l'infrastructure pour servir les développeurs sont les mots d'ordre des blips que nous présentons ici.

PAR **TÉRENCE CHATEIGNÉ**
Lead Ops

ADOPT
1/5 blips

50 ArgoCD

ArgoCD est une plateforme de Continuous Delivery (CD) GitOps qui permet de déployer des applications dans des clusters Kubernetes, de manière déclarative.

#GitOps

ArgoCD est un outil open source de Continuous Delivery (CD) pour Kubernetes, basé sur une approche GitOps. **Il déploie des ressources dans des clusters, en utilisant un ou plusieurs repositories Git comme source de vérité**. L'outil utilise une Custom Resource appelée Application, où l'on définit des paramètres tels que le repository source, le namespace, le cluster Kubernetes de destination, et la stratégie de réconciliation. ArgoCD supporte plusieurs types de sources pour déployer des ressources dans Kubernetes : charts Helm, applications Kustomize, fichiers Jsonnet ou encore de simples manifestes.

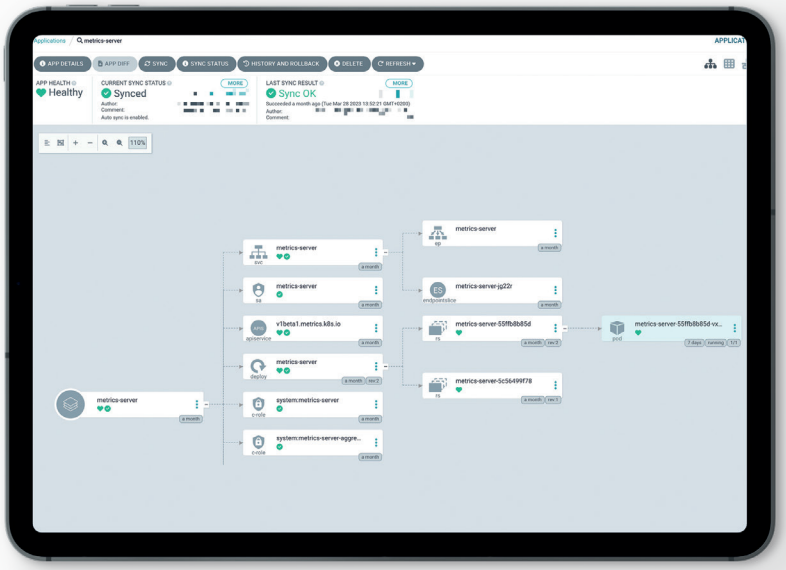
Pour subvenir à des besoins plus avancés, la Custom Resource ApplicationSet et son contrôleur associé permettent de générer plusieurs Applications ArgoCD à partir d'un unique manifeste. En utilisant les generators et les templates d'ApplicationSet, il est alors possible de **déployer plusieurs applications dans plusieurs clusters Kubernetes**, tout ceci dans une spécification centralisée !

C'est une plateforme qui trouve son intérêt dans le fait que la gestion des applications et des ressources dans Kubernetes peuvent mener à des erreurs humaines ou des incohérences. Par son fonctionnement, ArgoCD **se veut simple à utiliser et à configurer : ce qui est défini dans votre repository Git représente ce qui est déployé dans votre cluster Kubernetes**. En effet, il bénéficie d'une fonctionnalité qui lui **permet de vérifier l'état des ressources à intervalles réguliers**, afin de le **réconcilier automatiquement** si nécessaire. Par conséquent, c'est un outil sécurisant tant pour les développeurs que pour les administrateurs.

ArgoCD se veut aussi modulaire : vous pouvez gérer et mettre à jour automatiquement les versions d'images de conteneurs dans votre cluster avec ArgoCD

Image Updater, suivant les principes GitOps. Avec ArgoCD Notifications, vous obtenez du monitoring et de l'alerting sur les déploiements d'applications, bien que cette fonctionnalité soit encore immature.

La force d'ArgoCD réside dans son **interface utilisateur**, qui permet en un clin d'œil de comprendre l'état actuel des ressources, leur statut de synchronisation avec le repository source ou encore l'état des Pods associés à l'Application. En quelques clics seulement, nous arrivons vite à comprendre que cette Application déploie plusieurs ressources, comme un Service ou un Deployment. À son tour, un Deployment provoque la création d'un ou plusieurs Pods, ce qui est également pris en compte dans l'interface.



Cette vue donne de façon synthétique une arborescence des ressources Kubernetes déployées par une Application ArgoCD.

NOTRE POINT DE VUE

Pour ces raisons, ArgoCD est devenu, chez Theodo Cloud, le standard vers lequel se tourner lorsqu'il s'agit de déployer des applications dans Kubernetes. Chez nombre de nos clients, l'**implémentation d'ArgoCD a permis aux développeurs de mettre un pied dans le monde de l'infrastructure**, en utilisant un environnement rassurant, qui propose une gestion simplissime au quotidien.

ADOPT
2/5 blips

51 ELK (Stack ELK)

La stack ELK (Elasticsearch Logstash Kibana) propose une solution complète pour la gestion des logs et des performances de vos applications.

#Observabilité

La stack ELK (Elasticsearch, Logstash, Kibana) est la suite populaire d'outils open source, développée par l'équipe Elastic.

Elle **permet de stocker, indexer, visualiser et analyser les logs et performances de vos applications et de vos outils**.

La stack permettra de répondre à plusieurs des enjeux des architectures et des applications modernes :

- Centralisation des logs applicatifs et d'infrastructure ;
- Visualisation en temps réel ;
- Suivi des performances de vos applications et de vos outils (APM) ;
- Mise en place de dashboards métiers avec Kibana ;
- Détection d'anomalies et mise en place d'alerting.

La suite peut être complexe à déployer et à maintenir. Une bonne expertise des différents outils est nécessaire sur le long terme pour mettre à jour et faire évoluer la plateforme par rapport à vos besoins.

Outre le coût de la maintenance, la suite complète pourra demander beaucoup de ressources (CPU, RAM et espace disque) en fonction de la quantité (historique) de logs que vous allez traiter.

Dans ce contexte, **on ne saurait trop vous recommander d'utiliser l'opérateur Kubernetes ELK pour déployer la stack**.

La gestion n'en sera que facilitée, notamment pour des déploiements modestes.

Sur le long terme, vous devrez vous appuyer sur des compétences techniques solides et des connaissances approfondies sur chaque composantes. Il est essentiel de bien comprendre comment configurer des paramètres avancés pour la gestion de vos logs (rotation, sauvegarde, purge, etc).

Cependant, investir du temps et des efforts pour maîtriser la stack ELK en vaut véritablement la peine. Une fois en place, c'est un outil extrêmement puissant qui **améliorera l'expérience et l'efficacité de vos développeurs**. Ces derniers pourront alors se concentrer sur la production d'applications de meilleure qualité, plus stables et plus performantes.

NOTRE POINT DE VUE

La stack ELK est un outil puissant, exigeant techniquement, qui **tiendra toutes ses promesses** à long terme, à condition d'avoir les compétences techniques nécessaires. C'est un choix plus flexible et évolutif que les outils natifs des Cloud Providers.

ADOPT
3/5 blips

52 Helm

Helm est une solution de packaging pour le déploiement d'applications conteneurisées dans Kubernetes.

Lorsque vous utilisez Kubernetes pour déployer vos applications, le nombre de manifestes (fichiers de configuration de ressources Kubernetes) augmente rapidement. **Cela entraîne plusieurs problèmes :**

- Gérer manuellement le code et les dépendances applicatives (comme Redis, une BDD ou un reverse proxy) devient volumineux et coûteux à maintenir.
- Versionner un ensemble de ressources Kubernetes en une seule application est très complexe.
- Déployer une même application dans plusieurs environnements va demander de la duplication de code.

Helm répond à tous ces problèmes. Son atout réside dans l'utilisation du moteur de templating du langage Go. Grâce à ce dernier, la création de manifestes Kubernetes est transformée en l'application de **values** (variables dépendant du contexte de déploiement) à des **templates** (manifestes Kubernetes agrémentés de balises de templating).

L'ensemble des templates et du fichier de values par défaut constitue un **chart**. Un chart est versionné et peut être déployé dans un cluster Kubernetes (on déploie alors une release Helm). D'autres charts peuvent être déclarés en tant que dépendances, permettant ainsi de déployer des stacks applicatives complètes.

La CLI permet de déployer des charts locaux ou distants, étant donné que Helm apporte la possibilité de créer des **registries de charts**. Elle permet de déployer des versions précises, d'effectuer des mises à jour ou des rollbacks de ses applications.

Nous utilisons des charts pour déployer Prometheus/Grafana, Cert-Manager, Cluster Autoscaler ou Nginx Ingress Controller entre autres.

NOTRE POINT DE VUE

Chez Theodo Cloud, nous avons créé une librairie de charts Helm pour les applications que l'on utilise le plus chez nos clients. Elle raccourcit de journées entières la mise en place d'ensembles complexes d'applications. Helm est donc un outil que nous recommandons à tous les utilisateurs de Kubernetes.

ADOPT
4/5 blips

53 Integrated CI/CD

La CI/CD intégrée regroupe les services tels que GitHub Actions et GitLab CI. Ils sont directement incorporés dans les plateformes éponymes, au plus proche du code.

#CI/CD

La CI/CD (Continuous Integration / Continuous Delivery ou Continuous Deployment) représente **les pratiques d'automatisation des étapes de mise en production d'une application** : les tests, le build, la release et le déploiement. Des outils comme Jenkins, CircleCI et TeamCity permettent de créer des pipelines CI/CD, et sont dits "externes" car ils n'hébergent pas le code source de l'application sur laquelle les pipelines se basent. **GitHub et GitLab sont les principales plateformes de développement Git collaboratif**, avec environ 130 millions d'utilisateurs combinés. Les développeurs les utilisent quotidiennement pour coder, approuver des pull requests et gérer des issues.

Ces deux plateformes proposent leurs propres moteurs de CI/CD : GitHub Actions et GitLab CI. Grâce à des fichiers YAML, vous pouvez rapidement créer des pipelines d'automatisation qui optimisent le temps de développement. Leur free tier est généreux, évitant le besoin de payer ou de créer une infrastructure dédiée pour les petites équipes. **Ces outils offrent probablement toutes les fonctionnalités nécessaires**, mais des abonnements premium sont disponibles pour des besoins spécifiques ou des fonctionnalités avancées.

Au-delà de ça, **les deux solutions sont auto-hébergeables**. Le système de runners de GitLab CI dans Kubernetes est particulièrement bien rôdé. Il rend possible **la construction rapide d'une plateforme d'exécution de jobs** extensible à l'infini. Cependant, nous vous mettons en garde contre l'hébergement de vos runners si vos besoins ne s'y prêtent pas. Les coûts en maintenance et en calcul peuvent exploser. En outre, en tant que fonction clé du cycle de vie d'une application, une CI/CD instable peut rendre très difficile la vie de vos développeurs.

NOTRE POINT DE VUE

Malgré cela, nous recommandons à tous nos clients de se tourner vers l'une de ces deux plateformes pour créer leur CI/CD. Leur intégration au plus près du code les rend extrêmement versatiles et réduit énormément la boucle de feedback.

ADOPT
5/5 blips

54 Sentry

Sentry est un outil puissant pour suivre et corriger les erreurs applicatives.

#Observabilité

Sentry aide les développeurs à suivre les erreurs applicatives grâce à **une interface web intuitive**. En installant le SDK adapté à leur langage de programmation (Node.js, Python, Ruby, Go, Android, etc.), ils capturent les erreurs et exceptions. Cela leur permet d'obtenir **des informations détaillées sur la cause première de chaque erreur**, améliorant ainsi la qualité des services.

Depuis quelque temps, Sentry gagne en popularité en raison de sa **capacité à fournir des informations exploitables** pour améliorer la qualité du code et la fiabilité des applications. En plus de suivre les erreurs, Sentry permet de **surveiller les métriques de performance des applications en temps réel**, aidant ainsi à identifier les points de défaillance uniques (SPOFs) et les problèmes de scalabilité.

Pour intégrer Sentry dans votre écosystème, **vous avez deux possibilités** : utiliser **la solution SaaS** (payante) ou bien l'installer en **self-hosted** sur votre infrastructure, comme un cluster Kubernetes. **Nous vous recommandons d'utiliser la solution SaaS** qui est généralement plus rentable à long terme en matière de temps consacré à la maintenance.

En effet, **l'architecture applicative de Sentry est complexe** avec pas moins de 10 composants à gérer. Bien que Sentry puisse être déployé avec un Chart Helm développé par la communauté, maintenir un niveau de disponibilité satisfaisant en self-hosted nécessite beaucoup de temps et d'efforts pour corriger les erreurs natives du produit. Il est important de peser les avantages et les inconvénients de chaque approche selon vos besoins spécifiques.

NOTRE POINT DE VUE

Nous conseillons la solution SaaS de Sentry pour améliorer la qualité et la fiabilité des applications. Elle offre des mises à jour automatiques et un support direct de l'éditeur, évitant les complexités de la gestion en self-hosted. Mais pour certaines entreprises, cette dernière peut être plus adaptée pour des raisons de conformité ou de contrôle des données.

TRIAL
1/4 blips

55 GitHub Copilot

GitHub Copilot, est un outil de complétion de code basé sur l'IA, très utilisé par les Ops Theodo Cloud mais encore perfectible.

Intégrable facilement à un environnement de travail comme VSCode, GitHub Copilot est extrêmement puissant. Il permet de :

1. **COMPLÉTER**
   Copilot est très efficace pour créer rapidement des scripts Bash simples (usage que nous privilégions) et générer du code pour des langages tels que Go, Python, JavaScript, Java, ainsi que des templates basiques en Helm et Kubernetes.

   Cependant, sur des charts Helm complexes ou des ressources Terraform, il montre des limites. Les solutions proposées ne respectent pas toujours les bonnes pratiques que nous suivons chez Theodo Cloud, notamment en matière de sécurité, de principes DRY, et de lisibilité (WYSIWYG).

2. **EXPLIQUER**
   Grâce à la possibilité de prendre la codebase en contexte, **Copilot fournit des explications claires sur le code**. Cela s'avère très utile pour analyser une fonction ou naviguer dans un repository inconnu.

3. **CORRIGER**
   Copilot détecte rapidement les erreurs les plus évidentes, mais ne sait pas corriger les problèmes complexes impliquant plusieurs technologies. Par exemple, dans le cas d'ingress nginx mal configurés ou des jobs GitlabCI customs qui ne se lancent pas. **Nous retenons qu'il peut donner des pistes pour debug, mais rarement donner une solution complète**.

4. **DOCUMENTER**
   Nous avons déjà largement adopté Copilot pour produire de la documentation de qualité telle que : les swaggers, les catalogues Backstage, la description d'interfaces de Chart Helm ou encore l'explication de GitHub Actions.

NOTRE POINT DE VUE

En conclusion, GitHub Copilot est un outil que nous utilisons au quotidien mais qui **ne remplace pas encore nos SRE.** En attendant une amélioration du comportement de l'outil face à des problèmes complexes, nous le classons en **Trial**.

TRIAL
2/4 blips

56 Loki

Loki est l'outil de logging natif de la stack Grafana, idéal pour les environnements Kubernetes.

#Observabilité

Loki, développé par Grafana Labs, est **un outil de logging conçu pour collecter les logs de vos applications déployées dans Kubernetes et de les remonter dans Grafana**. Il permet de les interroger facilement via l'interface Web.

Cet outil de logging est **très simple à installer** dans vos clusters Kubernetes. Il est déjà packagé dans le chart prom-stack de la communauté : vous n'avez qu'une option à activer et le tour est joué, quasiment aucune configuration supplémentaire n'est requise.

Si vous utilisez déjà Grafana, vous pourrez créer des dashboards complets combinant les métriques et logs de vos applications, pour le monitoring et debugging. Rien de mieux pour **rendre vos développeurs complètement autonomes** dans la gestion de leur application !

Loki offre également la possibilité de mettre en place une rétention de logs sur le long terme en envoyant l'historique dans des buckets. Cependant, **l'outil n'est pas optimisé pour effectuer des recherches sur des logs archivés** : la recherche directe dans les logs accessibles dans votre cluster Kubernetes est plus performante que la recherche dans des logs archivés qui doivent être rapatriés, décryptés et indexés.

Inconvénient : le **langage qui sert à requêter les logs internes à Loki, le LogQL, n'est pas le plus facile à appréhender et maîtriser**. La syntaxe est différente des langages de requête traditionnels et nécessite de bien comprendre ce qu'est un range vector. Voici un exemple de requête LogQL pour illustrer cette différence :

```
{app="example-app"} |= "error"
```

NOTRE POINT DE VUE

Si vous utilisez déjà Grafana pour vos dashboards et cherchez une solution facile à mettre en place pour le logging et le debugging de vos clusters Kubernetes, Loki est un excellent choix. Cependant, gardez à l'esprit que Loki n'est pas optimisé pour la rétention à long terme de logs et que LogQL demande un peu d'apprentissage.

TRIAL
3/4 blips

57 Platform Engineering

Le Platform Engineering est une manière de concevoir son infrastructure comme un produit dont les développeurs sont les utilisateurs.

À l'origine, **le Cloud promettait de simplifier l'administration système** en abstrayant sa complexité. Mais la maturité du Cloud et l'augmentation des services en découlant (plus de 200 sur AWS), ont complexifié le paysage. La mouvance DevOps cherche à répondre à ce défi en intégrant des profils spécialisés pour apporter une expertise sans friction à l'équipe.

Pourtant, **le DevOps est difficile à faire passer à l'échelle** et les DevOps (re)deviennent le goulet d'étranglement du delivery :

- Le paysage des technologies et services se densifie, les pratiques et patterns évoluent.
- Pour que l'adage "You build it, you run it" reste vrai, les développeurs ont besoin d'expertises qui dépassent souvent leur champ de compétences.

Certaines Digital Factory abordent le Platform Engineering en produisant des outils communs pour toutes les business units de leur groupe. Cependant, **cette démarche exige un apprentissage de l'approche produit, complètement nouvelle pour une DSI historique**. L'impact ? Une plateforme très uniformisée et sécurisée, mais qui ne répond que partiellement aux besoins méconnus des développeurs.

Pour nous, le **Platform Engineering consiste à considérer l'infrastructure comme un produit à destination des développeurs**. Cela a 3 implications :

- collecter les besoins et feedbacks utilisateurs régulièrement, comme on le ferait si on lançait notre service de covoiturage ;
- déterminer les jobs to be done et performances critiques du produit à construire, ses personae, son architecture fonctionnelle ;
- repenser l'Operating Model. Notre recommandation : découper l'équipe DevOps en deux, une équipe Enabler livrant de nouveaux produits aux équipes tech, une équipe Operators assurant la fiabilité et la cohérence de la plateforme.

UN POINT D'ATTENTION

Le Platform Engineering est un vrai investissement de temps sur la démarche et sur le produit, il n'est donc intéressant qu'une fois passée une taille critique des équipes devs et ops.

TRIAL
4/4 blips

58 Preview Environments

Ils permettent de tester les changements dans un environnement réel réduisant le risque d'erreur et augmentant leur efficacité.

#DevX

Les preview environments permettent aux développeurs de **tester les changements sur une application avant le déploiement en production.** Utilisés pour vérifier les nouvelles fonctionnalités et les mises à jour de code, ils permettent notamment de tester l'interopérabilité des sous-systèmes et de s'assurer que tout fonctionne comme prévu. Cela réduit ainsi les risques d'erreur et le temps de correction.

Ces environnements améliorent aussi l'efficacité des développeurs qui peuvent travailler simultanément sur plusieurs fonctionnalités dans des environnements isolés. Cela diffère d'un environnement staging "classique".

Ces environnements peuvent être créés de différentes façons. **Voici la méthode que Theodo Cloud recommande aujourd'hui** :

1. À chaque ouverture de PR sur une application, on déploie, en plus de la version "stable", la version issue de la branche.
2. Pour chaque nouvelle version déployée d'une application, on définit un header qui permet de router des requêtes dessus.
3. Pour tester une PR, il suffit de faire des requêtes en utilisant ce fameux header.

Cela fonctionne parfaitement dans un cluster Kubernetes, mais risque d'être plus compliqué à mettre en place sans.

Toutefois, si cette méthode est très efficace pour les applications en HTTP, nous n'avons pas encore de solution à proposer pour les applications event-driven.

NOTRE POINT DE VUE

Les preview environments sont très utiles pour augmenter l'efficacité des développeurs de votre organisation. Cependant, ils sont un peu compliqués à mettre en place et à maintenir. Nous les recommandons donc principalement aux **équipes en grande expansion.**

ASSESS
1/3 blips

59 Argo Rollouts

Argo Rollouts permet facilement de mettre en place des modes de déploiement complexes (Blue-Green / Canary).

#CI/CD

Argo Rollouts, à l'instar de Flagger, est une solution permettant de faire ce qu'on appelle du "Blue-Green Deployment" ou des "Canary Release". Un "Blue-Green Deployment" implique deux environnements de production, tandis qu'une "Canary Release" déploie progressivement une nouvelle version à une fraction des utilisateurs.

Ces modes de déploiement complexes n'étaient pas accessibles à toutes les organisations avant la création de ces outils, car ils nécessitent le développement d'un outil custom ou la mise en place de workflows complexes.

Sa mise en place, comme celle de tous les outils de l'écosystème Argo, **passe par Kubernetes** et de nouvelles ressources (CRD Rollouts). L'outil tire judicieusement partie du support pour les subressources sur les CRD disponibles depuis Kubernetes 1.10 pour introduire une nouvelle ressource comparable en tout point à un Deployment (en utilisant la subressource / scale), mais en complétant les spécifications de la stratégie de mise à jour.

Ce faisant, vous pouvez désormais écrire des "**déploiements à étapes**" en utilisant notamment :

- Des étapes de scaling (augmentation du nombre de pods dans la nouvelle version) ;
- Des étapes de pause ;
- Une analyse en background pour déterminer si le déploiement doit continuer (par exemple une requête à Prometheus).

Pour les modes de déploiement complexes, nous privilégions Flagger que nous avons davantage testé dans nos projets.

NOTRE POINT DE VUE

Cependant, nous pensons qu'Argo Rollouts a un fort potentiel. Les deux outils tirent parti de l'intégration à leur écosystème respectif (FluxCD pour l'un, ArgoCD pour l'autre).

Mettre en place ces modes de déploiement exige une grande maturité et un très bon système d'observabilité.

ASSESS
2/3 blips

60 Backstage

Backstage est un outil couteau-suisse qui, si bien utilisé, peut devenir le point central de votre Internal Developer Platform (IDP).

#DevX

Backstage, technologie open source lancée par Spotify en 2020, vise à **créer un portail développeur extensible pour votre plateforme interne**.

L'idée derrière Backstage est de devenir le point de passage "obligatoire" et unique pour interagir avec votre plateforme interne. Dans sa version sans plugin additionnel, Backstage propose déjà plusieurs features intéressantes :

- **TechDocs**, pour agréger toutes les documentations au format Markdown dans votre instance Backstage ;
- **Software Templates**, pour la création de boilerplates afin de démarrer de nouveaux services (par exemple, l'initialisation du repository) ;
- **Software Catalog** pour référencer les différents services/ utilitaires que vous utilisez ou développez. Il peut par ailleurs agréger toutes les specs OpenAPI ou Swagger de vos services web.

La principale force de Backstage est son extensibilité. De nombreux plugins ont été développés par la communauté (par exemple une intégration avec ArgoCD) et il est tout à fait possible d'imaginer développer ses propres plugins afin de satisfaire les besoins internes.

Il ne convient cependant pas à toutes les organisations et n'est **réellement utile qu'à partir du moment où votre équipe Tech/R&D dépasse les 50 personnes**. Cette "taille critique" correspond au moment où suivre l'ensemble des évolutions de votre plateforme commence à être très complexe.

Le premier bénéfice de Backstage est de **faciliter l'onboarding des développeurs et DevOps** via la centralisation de la documentation technique. Ceci permet un gain important sur le KPI "Time to first PR" clé dans des équipes techniques conséquentes.

NOTRE POINT DE VUE

Nous sommes toujours en train d'observer les gains de Backstage sur des équipes tech de tailles différentes, et c'est pourquoi il reste pour le moment dans le cadran “Assess”.

ASSESS
3/3 blips

61 Kubernetes native CI/CD

Les outils de CI/CD Kube native sont totalement intégrés à Kubernetes.

#CI/CD #GitOps

L'extensibilité de Kubernetes via les CRDs (Custom Resource Definitions) a entraîné de nombreuses innovations dans divers domaines, y compris en CI/CD. Bien qu'ArgoCD puisse être considéré comme un outil de cette catégorie, il est aujourd'hui spécialisé dans le déploiement sur Kubernetes et ne couvre pas la partie CI (intégration continue).

Nous voyons donc émerger des **outils basés sur Kubernetes** qui se rapprochent des moteurs de workflow que nous utilisons/avons utilisé par le passé (e.g. Jenkins), permettant de définir de manière déclarative des pipelines de CI/CD ou tout autre workflow nécessaire.

Nous allons nous intéresser à 2 outils qui n'ont aujourd'hui pas tout à fait la même cible : **Tekton et Argo Workflow**.

Tekton est une extension (un opérateur) **de Kubernetes permettant de définir des workflows via de nouvelles ressources** (Task et Pipeline). Il permet également de **déclencher ces workflows via n'importe quel type d'événement** (EventListener et Trigger). Son intégration simple dans Openshift en fait un outil très utilisé pour la CI/CD et il peut être vu comme le successeur Kubernetes Native de Jenkins.

Argo Workflows, similaire à Tekton, permet d'accomplir à peu près la même chose. Cependant, il est principalement adopté par les équipes effectuant des **traitements de données (ETL) ou du machine learning**.

NOTRE POINT DE VUE

Nous ne sommes pas encore convaincus par ces approches pour remplacer les outils intégrés à vos providers git (e.g. Github Actions, Gitlab-CI) du fait de leur adoption déjà établie. Dans le cas où vous disposez d'une **infrastructure hétérogène** (e.g. VM-Based, Kubernetes etc.) ces outils vous permettront d'**homogénéiser votre manière de tester et de déployer.**

HOLD
1/2 blip

62 Fluxcd

Fluxcd est une solution de CD pour Kubernetes supportant les manifests Helm et Kustomize. Elle permet de déployer en se basant sur une source de configuration comme un dépôt Git.

#GitOps

L'absence d'interface graphique intégrée nativement à Fluxcd est un frein pour le suivi et le debug des déploiements. Son utilisation n'est pas à la portée de tout le monde, il faut bien connaître et comprendre Kubernetes pour pouvoir l'exploiter efficacement.

Pour déployer une seule application, **il faut créer au minimum 3 Custom Resources contre une seule avec ArgoCD**. Cette complexité augmente le temps pour lire une configuration de ces ressources, mais surtout pour comprendre le fonctionnement global de Fluxcd, ce qui contribue à diminuer l'accessibilité de l'outil.

Le controller de Fluxcd ne fait pas de réconciliation périodique entre le code et l'état existant. Il propose uniquement deux stratégies de réconciliation (**chartVersion** ou **revision**) qui met à jour les ressources lors d'un changement de la source (repository ou bucket). Ainsi, le code ne représente plus ce qui est déployé si un administrateur Kubernetes modifie des ressources manuellement. C'est contraire à un principe du GitOps : le code est la source de vérité.

Une solution de GitOps performante **doit rendre les développeurs autonomes** pour **déployer facilement** sur les environnements de développements et pour **mettre en production**. Quand les développeurs gagnent en autonomie, alors les Ops peuvent se concentrer sur des tâches à plus forte valeur ajoutée.

NOTRE POINT DE VUE

Fluxcd est une solution qui fonctionne et qu'on utilise en production chez certains de nos clients. Cependant, nous ne vous recommandons pas de l'installer, nos expériences avec l'outil nous poussant à **préférer ArgoCD.**

HOLD
2/2 blip

63 Jenkins

Jenkins est un orchestrateur de workflows très générique qui permet énormément de flexibilité, mais qui nécessite un gros effort de customisation.

#CI/CD

Jenkins est avant tout un orchestrateur de workflows automatisés. Il permet de déclarer des projets dans lesquels il est possible de définir des suites d'actions, que l'on peut ensuite exécuter. Il est également possible de visualiser l'évolution dans le temps de l'état ("success", "failure", etc).

L'interface tout comme les mécanismes disponibles pour les workflows sont **extrêmement personnalisables via des plugins**. Cela vous permettra d'utiliser des variables matricielles pour les actions, d'avoir une météo du projet, des tableaux de bord, voire de pouvoir lancer des scripts généralistes pour réaliser des actions sur vos plateformes de test.

Jenkins s'intègre aussi nativement avec le langage Groovy pour aller plus loin dans la définition des workflows.

Parmi l'ensemble des possibilités, on peut donc en particulier définir des **pipelines de CI/CD** via notamment des plugins d'intégration avec des gestionnaires de versions de code par exemple.

Cependant, les outils CI/CD spécifiques à certains environnements rendent un outil généraliste comme Jenkins moins attrayant, puisqu'il nécessite sa propre maintenance.

NOTRE POINT DE VUE

Nous recommandons de **privilégier les solutions CI/CD intégrées** (ex : GitHub Actions ou GitLab CI). Elles offrent toutes les fonctionnalités nécessaires au développement web moderne, tout en proposant une expérience clé en main. Cela nécessite beaucoup d'efforts de customisation initiale, mais offre plus de stabilité à l'usage.

Une équipe de choc

ALEXANDRE HERVÉ
Lead SecOps

AURÉLIEN VU NGOC
Lead Ops

BENJAMIN MARY
Ops

BENJAMIN SANVOISIN
Lead Ops

CLÉMENT DAVID
Co-founder & CEO

CLÉMENT DEBERDT
SecOps

FRANÇOIS BALIGANT
Principal Engineer

GUILLAUME LECCESE
Lead Ops

JOSÉPHINE SAINT-JOANIS
SecOps

KIMELYNE SERVAIS
Engineering Manager

LAURINE LEFORT
Digital Marketing Manager

LILA CHAUVEAU
Project Director

LUCAS MARQUES
Ops

PAUL VIOSSAT
Lead SecOps

PHILIPPE DUPART
Lead SecOps

STANISLAS BERNARD
CTO

TÉRENCE CHATEIGNÉ
Lead Ops

THIBAULT LENGAGNE
Head of Security

THIBAUT ROBINET
Lead Ops

XAVIER FINKELSTEIN
Lead Ops

Quelques chiffres clés

2009 Theodo est une entreprise internationale de conseil en technologie. Elle accompagne les entreprises innovantes dans la conception, le développement et le déploiement de produits digitaux ingénieux qui transforment la vie de leurs utilisateurs. Nos équipes rassemblent plus de 700 Theodoers passionnés de technologie et d'amélioration continue dans 5 pays en Europe, Afrique et Amérique du Nord. Nous sommes fiers de compter parmi nos clients BNP Paribas, Colas, ContentSquare, Dior, ManoMano, Safran et Qonto.

80 C'est le nombre de Theodoers qui travaillent aujourd'hui chez Theodo Cloud.

2018 Cofondée par Clément David et Aurore Malherbes, tous deux anciens de Theodo Apps, Theodo Cloud se concentre sur les problématiques cloud puis lance une division Cybersécurité en 2021.

+150 Projets cloud et DevOps co-construits avec des clients issus de grands groupes comme de l'univers start-up : BPI France, Meilleurs Agents, Mirakl, Sunday, Total et bien d'autres encore !

INFRASTRUCTURE & CLOUD

theodo.
Cloud