UBIKA
WAAP as a Container
Scalez votre protection sur Kubernetes

UBIKA WAAP as a Container est le nouveau WAAP ultra-scalable qui protège vos applications web et API depuis vos clusters kubernetes.

Les 3 défis clés du passage à la sécurité applicative distribuée

1

Continuité de service, un enjeu sous pression

La continuité de service est aujourd'hui sous tension face à la montée des attaques DDoS et à l'augmentation des coûts d'infrastructure, qui imposent aux entreprises d'être à la fois plus résilientes et plus efficaces. Elles doivent désormais s'appuyer sur des architectures capables d'absorber des pics de trafic tout en optimisant les ressources, ce qui favorise l'adoption de modèles élastiques. Dans ce contexte, Kubernetes s'impose comme un standard, en permettant d'automatiser le déploiement, la scalabilité et la résilience des applications, et devient ainsi un pilier essentiel de la disponibilité des services modernes.

2

Adapter la sécurité aux microservices déjà en production

La réalité opérationnelle de la plupart des organisations est que leurs clusters K8s sont déjà en production, les architectures microservices sont déployées, et les pipelines CI/CD tournent. La sécurité doit donc souvent s'adapter à l'existant.

3

Intégrer le WAAP dans les workflows DevSecOps existants

Face à l'évolution rapide des architectures et des menaces, les organisations intègrent la sécurité directement dans le cycle DevSecOps afin de réduire les erreurs humaines et renforcer la gouvernance. Les politiques de sécurité deviennent du code versionné dans Git, soumis à review, visible et traçable, garantissant un principe de "no drift" où l'état réel reste aligné avec l'état défini. Dans Kubernetes, cette approche se concrétise via le policy as code et l'automatisation via les pipelines CI/CD.

UBIKA WAAP as a Container : la réponse aux défis modernes

Avec **UBIKA WAAP as a Container**, gérée via des CRDs, la protection applicative devient **Security & workflow as code by design**. La sécurité et les processus opérationnels sont traité comme du code, conçu dès l'origine pour être versionné, automatisé et intégré dans les pipelines, garantissant cohérence, gouvernance et scalabilité.

1

Une politique de sécurité versionnée et gouvernée par Git

UBIKA WAAP as a Container permet de définir la politique de sécurité sous forme déclarative, typiquement en YAML, directement dans les repositories Git des applications ou de la plateforme.

Concrètement :
- Les règles de sécurité deviennent des artefacts versionnés
- Chaque modification est historisée
- L'ensemble des politiques est lisible, auditable et traçable

Bénéfices :
- Transparence totale : Pas de règles cachées dans une console ou un équipement
- Audit facilité : qui a modifié quoi, quand et pourquoi
- Alignement Dev / Sec / Ops : un même référentiel de vérité

2

Des changements soumis à des revues comme du code applicatif

Avec **UBIKA WAAP as a Container**, toute évolution de la politique de sécurité peut être soumise à un processus standard :
- Pull request / merge request
- Relecture par des pairs (Dev, Sec, Platform)
- Validation avant mise en production

Pour concrètement :
- Ajouter une règle anti-bot
- Modifier un rate limit
- Activer une protection API

Bénéfices :
- Réduction du risque d'erreur humaine
- Cohérence des règles dans le temps
- Responsabilisation des équipes

3 Une capacité de rollback immédiat en cas d'incident

Dans un contexte de production critique, un changement de règle de sécurité peut avoir des effets non attendus comme les faux positifs bloquants, dégradation de la disponibilité, impact business.
Avec **UBIKA WAAP as a Container** un simple revert Git permet de revenir à un état fonctionnel connu.

Bénéfices :
- Retour arrière instantané
- Réduction du MTTR (Mean Time To Repair)
- Sécurisation des mises en production

4 Une garantie de "no drift" entre l'état désiré et l'état réel

Dans les systèmes traditionnels, les configurations dérivent. Il y a des modifications manuelles en production, des correctifs urgents non documentés et des incohérences entre environnements.
UBIKA WAAP as a Container est intégré dans une logique Kubernetes / GitOps :
- Considère Git comme la source de vérité
- Applique en continu l'état désiré
- Détecte toute divergence

Bénéfices :
- Aucune dérive silencieuse
- Alignement permanent entre code et réalité
- Sécurité reproductible entre environnements

5 Respect des design patterns Kubernetes

UBIKA WAAP as a Container est conçu pour fonctionner nativement dans Kubernetes, en respectant ses standards :
- Déploiement via Helm charts
- Intégration avec Ingress Controllers
- Possibilité de fonctionnement en sidecar ou service
- Configuration via CRDs (Custom Resources)

6 Une meilleure scalabilité, alignée avec les workloads

Grâce à cette intégration cloud-native :
- Le WAAP scale avec les Pods / services
- Pas de point unique de congestion
- Adaptation automatique aux pics de trafic (ex : DDoS L7)

Bénéfices
- Scalabilité horizontale automatique
- Performance cohérente avec l'application
- Coûts optimisés (pas de surdimensionnement)

7 Une observabilité intégrée dans l'écosystème existant

UBIKA WAAP as a Container s'intègre parfaitement aux outils standards Kubernetes :
- Prometheus pour l'export métriques
- Grafana pour la visualisation
- Pipeline centralisé (ELK, SIEM…) pour la gestion des logs

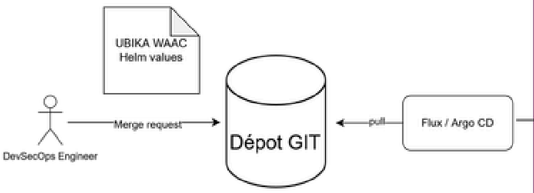
Exemples de métriques :
- Requêtes bloquées
- Types d'attaques
- Endpoints ciblés
- Anomalies comportementales

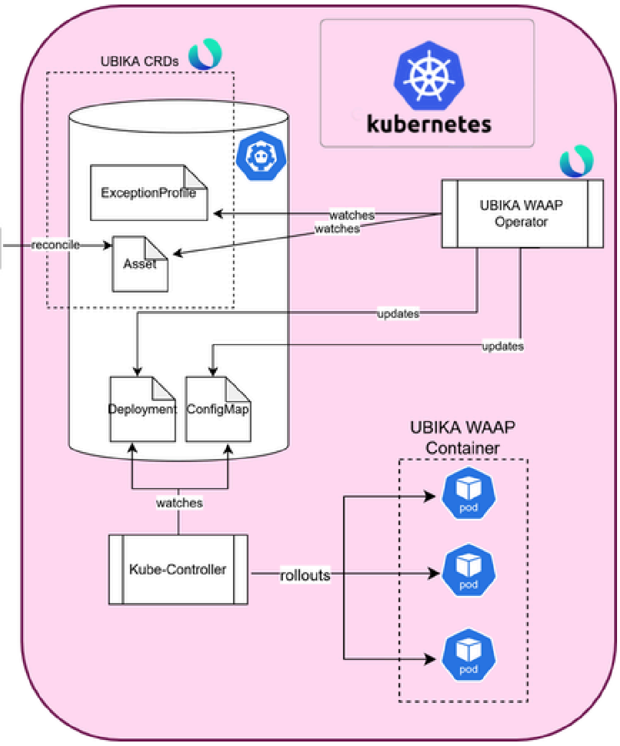
Bénéfices :
- Visibilité unifiée app + sécurité
- Corrélation rapide en cas d'incident
- Intégration sans surcouche tooling

8 Protection de vos applications web et API hors microservices et kubernetes

UBIKA WAAP as a Container protège également les applications qui ne sont pas des microservices ou ne sont pas dans kubernetes. Le nettoyage du trafic en amont permet à vos apps de ne pas avoir à consommer excessivement la bande passante. Ainsi, **UBIKA WAAP as a Container** donne à votre infrastructure la possibilité d'évoluer en douceur et en toute sécurité vers kubernetes.

Comment ça marche ?





- ✓ **Security & workflow as code by design**
- ✓ La politique de sécurité est versionnée, lisible, protégée par Git
- ✓ Tout changement de la politique peut être soumis à review
- ✓ En cas de panique, un revert de commit remet dans l'état initial
- ✓ Respect des design patterns de k8s : meilleure scalabilité, observabilité

UBIKA est une marque déposée par la société UBIKA. Toutes les autres marques mentionnées dans le présent document appartiennent à leurs propriétaires respectifs.
Version 01.00 | Juin 2026
Données sous réserve de modification
©2026 UBIKA | France