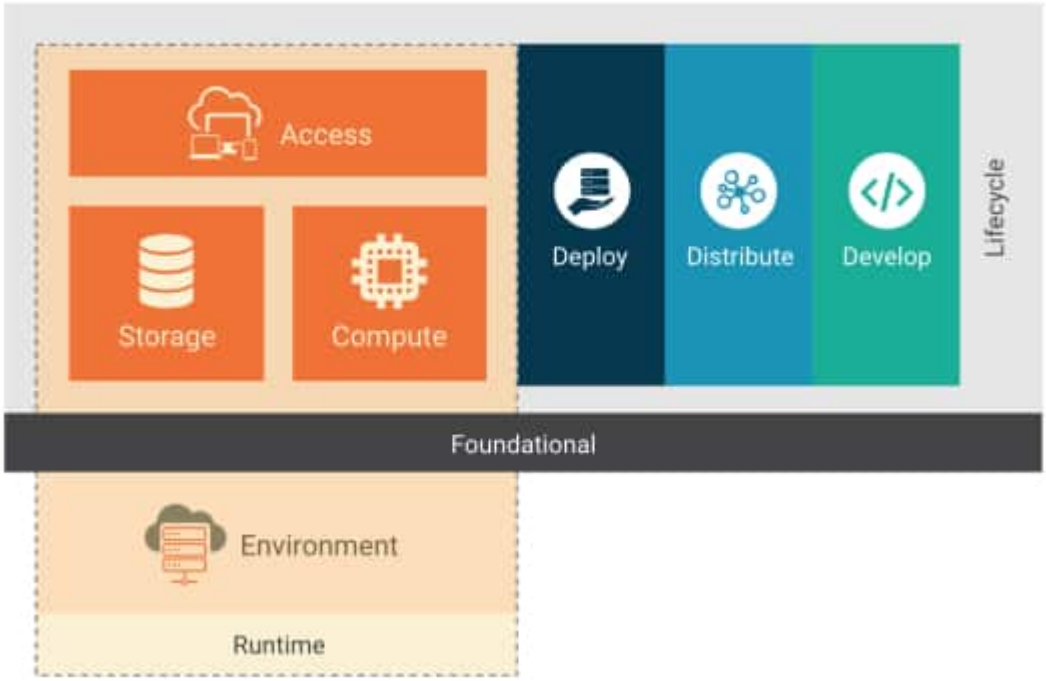
Développement cloud natif : comment gérer la sécurité ?

Connaissez-vous la fondation OWASP, le *framework* Att@ck, le principe de la « pyramide de test » et la règle des « quatre yeux » ? La CNCF les évoque tous dans un guide – en anglais – sur la sécurité des architectures *cloud-native*. Elle en donne la définition suivante.

> *Les technologies nativement cloud permettent aux entreprises de construire et d'exploiter des applications élastiques dans des environnements modernes et dynamiques comme des clouds publics, privés ou bien hybrides. Les conteneurs, les services maillés, les micro services [sic], les infrastructures immuables et les API déclaratives illustrent cette approche.*
>
> *Ces techniques permettent la mise en œuvre de systèmes faiblement couplés, à la fois résistants, pilotables et observables. Combinés à un robuste système d'automatisation, ils permettent aux ingénieurs de procéder à des modifications impactantes [sic], fréquemment et de façon prévisible avec un minimum de travail.*

Le guide se structure en quatre grandes parties. Lesquelles correspondent à autant de phases du « cycle de vie » des applications : développement, distribution, déploiement et exécution.

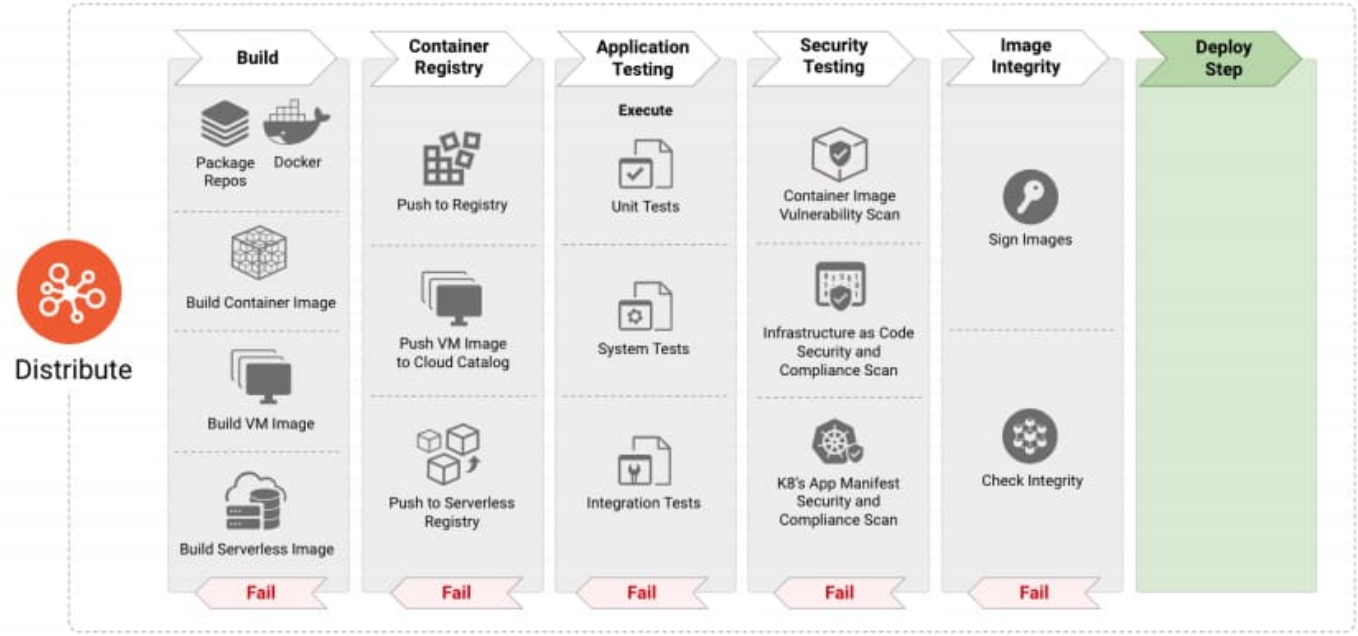


Quatre, c'est aussi le nombre de modèles d'exploitation *cloud-native* que distingue la CNCF. Parmi eux, il y a le IaaS et le PaaS. Mais aussi le CaaS (conteneurs en tant que service) et le FaaS (fonctions en tant que service). La fondation affirme que l'un et l'autre sont porteurs de bénéfices en matière de sécurité… à condition que celle-ci soit bien implémentée. Au vu de la pénurie de compétences, elle estime que les fournisseurs cloud ont une vraie carte à jouer sur leurs offres managées.

DevOps et dépendances

Tout en amont du cycle de vie, la CNCF insiste sur la nécessité d'intégrer la sécurité aux démarches DevOps. On s'assurera, par exemple, que l'analyse des manifestes d'applications et des modèles d'infrastructure en tant que code se fasse dans les IDE ou lors des requêtes *pull*.
C'est sur cette phase qu'interviennent le principe des « quatre yeux » et la fondation OWASP. Le premier suppose, dans les grandes lignes, une double révision du code. La seconde est mentionnée pour sa méthode de modélisation des menaces.

Pour la phase de distribution, l'attention se porte sur les dépendances… et sur les risques qu'elles impliquent. La CNCF revient sur les enjeux associés aux *pipelines* d'intégration continue : isolation des serveurs, protection des secrets, renforcement incrémental, application des règles de conformité, etc.

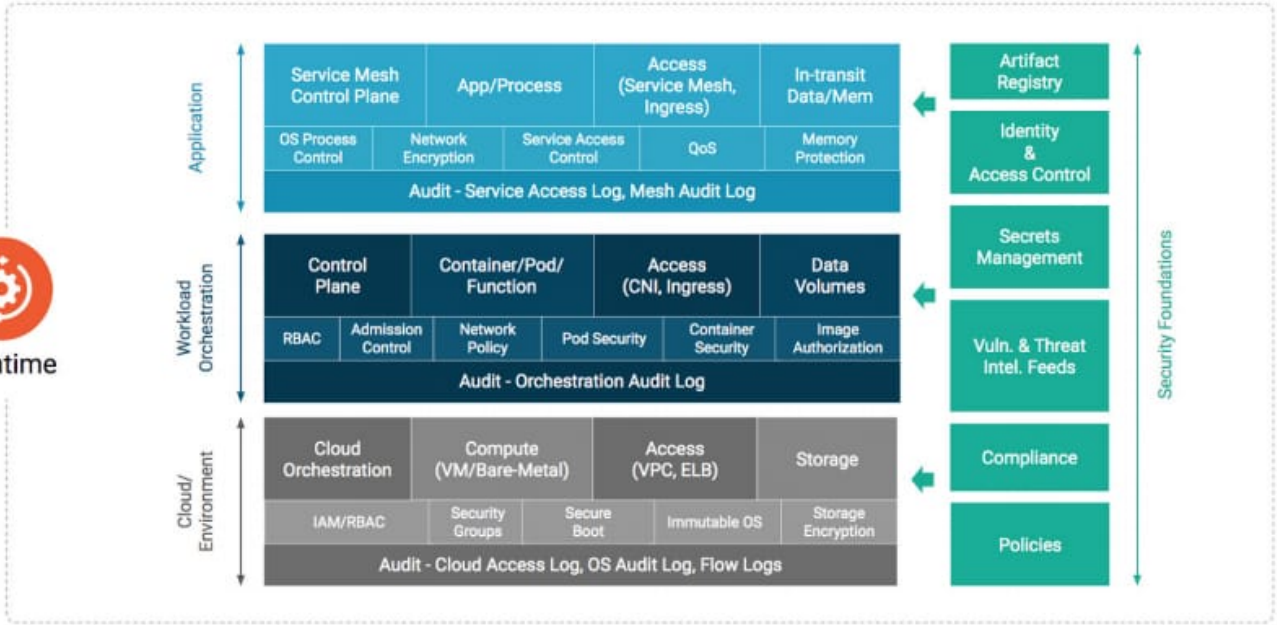


L'apport de l'IA – ou tout du moins l'analyse algorithmique – se ressentira particulièrement en phase de déploiement. C'est à ce moment-là qu'on vérifiera, entre autres, l'intégrité des images et leur politique d'exécution, ainsi que la conformité des hôtes.

Conteneurs : des couches supplémentaires à sécuriser

Pour la partie *runtime*, la CNCF concentre son discours sur les conteneurs. Elle rappelle, entre autres, l'importance d'utiliser un OS spécifique, en lecture seule et sans services superflus. Si possible en combinaison avec un (v)TPM, pour sécuriser l'ensemble de la plate-forme.

Il y a effectivement beaucoup de couches à surveiller. L'orchestrateur en est une. Il est vulnérable aussi bien à travers son API que son tableau de bord ou son magasin clé-valeur. Solution, d'après la CNCF : appliquer le principe du moindre privilège. « Peut-être l'aspect le plus important des architectures *cloud-native* », à prendre en compte « à tout endroit de la pile où se joue une décision d'authentification ou d'autorisation », explique-t-elle.



Parmi ses autres conseils, on aura relevé :

- Transmettre immédiatement les *logs* vers un endroit inaccessible avec les identifiants du cluster (pour éviter qu'un attaquant dissimule ses traces)
- Configurer l'authentification mutuelle entre tous les éléments du plan de contrôle
- Gérer les secrets avec un magasin externe fondé sur un HSM – ou, à défaut, en confier la gestion à l'orchestrateur

Pour le FaaS en particulier :

- Créer une liste blanche de fonctions que les processus peuvent exécuter
- Empêcher ces fonctions de réaliser des changements sur des points de montage critiques du système de fichiers
- Limiter leur accès aux services et surveiller leur trafic sortant

Illustration principale © Rawpixel.com – stock.adobe.com