FLASHMODZ ENTERPRISE

Cabinet d’expertise IT — Charleroi · Belgique & international

FlashModz

Infrastructure as Code pour Indépendants en Belgique

Guide IaC / Indépendants

Vous gérez vos serveurs à la main, votre config n'est documentée nulle part, et si votre laptop meurt demain, vous reconstruisez tout de mémoire. Ce guide vous montre comment automatiser votre infrastructure, même en solo.

Scannez pour acceder
a la version en ligne

contact@flashmodz.be | +32 471 87 87 07

https://flashmodz.be

L'essentiel en 5 minutes

Si vous gérez vos serveurs à la main, vous accumulez une dette qui finira par vous rattraper.

1 Le travail manuel sur serveur est du temps non facturable

Chaque déploiement à la main, chaque mise à jour risquée, chaque dépannage de config prend des heures que vous ne facturez pas. Et ce temps revient à chaque projet, parce que rien n'est réutilisable. Codifié une fois, un setup se rejoue à volonté.

2 Vos serveurs sont des "boîtes noires", un risque business

Sans IaC, votre setup est une boîte noire dans votre tête. Si demain vous tombez malade, partez en vacances ou décidez de vendre votre activité, personne ne peut reprendre. C'est une fragilité que vos clients ignorent, mais qui les expose autant que vous.

3 3 outils suffisent pour l'essentiel des bénéfices

Git pour versionner vos configs, Ansible pour automatiser le setup, Docker Compose pour vos environnements. Ces 3 outils gratuits, abordables en quelques semaines d'apprentissage progressif, suffisent à transformer un freelance "artisan" en freelance "industriel". Le retour se sent dès les premiers projets répétés.

4 L'IaC est aussi un argument commercial

Les clients qui choisissent leurs prestataires regardent désormais la maturité technique. Pouvoir dire "votre setup est entièrement codifié, reproductible, auditable, et un autre prestataire peut le reprendre" vous distingue de la masse des freelances. C'est un argument de continuité qui pèse dans une négociation.

5 C'est aussi un sujet de conformité, pas seulement de productivité

Le RGPD (art. 32) exige de pouvoir démontrer les mesures techniques protégeant les données personnelles que vous hébergez pour vos clients. Un dépôt Git avec vos playbooks EST cette démonstration : configuration du firewall, chiffrement, gestion des accès, tout est écrit et daté.

Et maintenant ?

Les pages qui suivent détaillent chacun de ces points avec exemples, chiffres et actions concrètes.

Document conçu pour Indépendants & Freelances Tech · Lecture : ~15-20 min · Édition 2026

Table des matières

Naviguez rapidement vers les sections qui vous concernent.

Le marché belge en chiffres

Chapitre 1

0 €

Git, Ansible et Docker Compose sont gratuits et open source : le ticket d'entrée de l'IaC est nul

Sources : textes officiels (NIS2, RGPD, DORA) et publications des éditeurs cités

1 fichier

un playbook Ansible versionné documente et rejoue l'installation complète d'un serveur

100×

un playbook idempotent peut être rejoué sans risque : il n'applique que ce qui manque

2021

l'incendie du datacenter OVH de Strasbourg a rappelé qu'une sauvegarde au même endroit n'est pas une sauvegarde

Contexte & enjeux

Soyons honnêtes : en tant qu'indépendant dans la tech, vous avez probablement un VPS chez Hetzner ou OVH, un ou deux projets en production, et une configuration que vous êtes le seul à comprendre.

Vous vous connectez en SSH, vous modifiez des fichiers à la main, vous installez des paquets au fil de l'eau. Ça fonctionne, jusqu'au jour où ça ne fonctionne plus. Un serveur qui plante, un upgrade qui casse tout, un client qui a besoin du même environnement et vous passez 3 jours à essayer de reproduire ce que vous aviez fait en 6 mois.

L'Infrastructure as Code (IaC), ce n'est pas un truc de grand groupe avec des équipes DevOps de 50 personnes. C'est une philosophie : au lieu de configurer votre infrastructure à la main, vous la décrivez dans des fichiers. Ces fichiers, vous les versionnez dans Git, vous les rejouez à volonté, et votre infra est documentée par définition. Deux propriétés font toute la différence : l'idempotence (un playbook peut être rejoué sans casser ce qui existe) et la reproductibilité (un serveur perdu se reconstruit depuis le code, pas de mémoire).

L'incendie du datacenter OVH à Strasbourg en 2021 l'a montré brutalement : des milliers de clients ont découvert le même jour que leurs sauvegardes vivaient au même endroit que leurs serveurs, et que la reconstruction reposait sur leur seule mémoire. Un playbook Ansible versionné et des sauvegardes externalisées changent ce scénario du tout au tout : quelques heures au lieu de plusieurs jours.

Chez FlashModz Enterprise, on pratique l'IaC au quotidien sur des infrastructures de toutes tailles. Ce guide vous montre comment commencer, avec les bons outils, sans sur-ingénierie. Adapté à votre réalité de freelance, pas à celle d'une licorne de la Silicon Valley.

> *"L'IaC, ce n'est pas un luxe de grand groupe. C'est ce qui vous fait passer du rôle d'artisan qui résout des problèmes à celui d'ingénieur qui les empêche d'arriver."*
>
> — Jérémy Manet, FlashModz Enterprise

Les risques d'une infrastructure non codifiée

Chapitre 2

Ces problèmes coûtent cher chaque jour aux indépendants belges. Et ils sont évitables.

Votre serveur plante. Votre documentation, c'est votre mémoire.

Pas de runbook, pas de script de déploiement, pas de fichier de config versionné. Vous reconstruisez tout de tête, en espérant ne rien oublier. Comptez plusieurs jours de reconstruction pour un serveur qui héberge des sites clients. Pendant ce temps, vos clients attendent, et certains ne reviennent pas.

"Ça marchait sur mon serveur de dev"

Votre environnement de développement et votre production sont configurés différemment sans que vous le sachiez. Un package en version X ici, version Y là-bas. Résultat : des bugs fantômes qui n'apparaissent qu'en prod, souvent devant un client. Docker Compose élimine ce problème en figeant les versions dans un fichier.

L'upgrade qui casse tout un vendredi soir

Vous faites un "apt upgrade" et votre stack LAMP s'effondre. La nouvelle version de PHP n'est pas compatible avec votre CMS, nginx a changé sa syntaxe de config. Sans IaC, il n'y a pas de rollback propre : juste de la sueur froide et du debug nocturne.

Le client qui veut "exactement la même chose"

Un nouveau client veut le même setup que votre autre client. Vous passez 2 jours à tout refaire à la main, et forcément, il y a des différences. Avec l'IaC, vous rejouez un playbook avec de nouvelles variables et c'est identique, en une fraction du temps.

Mythes vs Réalité

Chapitre 3

Six idées reçues qui empêchent d'agir — et ce qu'en disent les chiffres.

01

ON ENTEND SOUVENT

Je suis seul, j'ai 3 serveurs, l'IaC c'est pour les grosses équipes DevOps.

EN RÉALITÉ — *Conçu pour être accessible*

Ansible et Docker Compose sont conçus pour être accessibles. Avec quelques semaines d'apprentissage progressif, vous automatisez votre setup type. Le retour sur temps investi se mesure dès les premiers projets clients répétés. Plus vous attendez, plus la dette s'accumule.

02

ON ENTEND SOUVENT

J'ai des scripts bash qui automatisent mes installations, c'est pareil.

EN RÉALITÉ — *Idempotence : propriété clé*

Les scripts bash ne sont pas idempotents : les relancer peut casser un système déjà configuré. Ils ne gèrent pas les états, ne tracent pas les changements, et deviennent illisibles en grossissant. Ansible apporte l'idempotence, la lisibilité YAML et des milliers de modules éprouvés, pour un effort d'apprentissage raisonnable.

03

ON ENTEND SOUVENT

Je n'ai jamais le temps d'apprendre l'IaC entre deux missions client.

EN RÉALITÉ — *Le 1er playbook est un asset*

Commencez par codifier votre prochaine installation client au lieu de la faire à la main. Vous prenez un peu plus de temps la première fois, mais vous obtenez un asset réutilisable pour tous les clients suivants. Dès les projets similaires suivants, le gain devient net.

04

ON ENTEND SOUVENT

OVH fait des snapshots de mes VPS, je suis tranquille en cas de problème.

EN RÉALITÉ

Vendor lock-in vs portabilité

Les snapshots hébergeur sauvent l'image du serveur, pas la connaissance de comment il a été configuré. Et ils vivent chez le même hébergeur que le serveur, dans le même incident. Avec un playbook Ansible + Git, vous redéployez chez n'importe quel autre hébergeur, sans dépendre de personne.

05

ON ENTEND SOUVENT

Mes clients ne savent même pas ce qu'est l'IaC, ça ne leur apporte rien.

EN RÉALITÉ

Différenciation marché

Vos clients ne savent pas ce qu'est l'IaC, mais ils valorisent la rapidité, la fiabilité et la transparence. Pouvoir dire "je peux te livrer un environnement de staging en 10 minutes" est un argument commercial puissant. Et la traçabilité Git est un argument RGPD réel.

06

ON ENTEND SOUVENT

Docker, Kubernetes, c'est de la sur-ingénierie pour un freelance.

EN RÉALITÉ

Docker Compose ≠ Kubernetes

Docker Compose, c'est un fichier YAML de 30 lignes pour orchestrer une app + une base + un cache. Maîtrisable en un weekend. Kubernetes, c'est effectivement excessif pour un freelance, mais Docker Compose est dimensionné pour vous. La distinction est importante.

Cas concrets en Belgique

Chapitre 4

Trois scénarios types, inspirés de situations réelles et anonymisés.

CAS 1 // Dev web à Namur — 4 jours pour reconstruire un serveur

Contexte: Sébastien héberge 12 sites clients sur un VPS chez OVH. Pas de documentation, pas de scripts. Quand OVH Strasbourg prend feu en 2021, ses sauvegardes étaient sur le même datacenter.

Impact: 4 jours de reconstruction à la main. 2 clients perdus. 3 sites partiellement récupérés via la Wayback Machine. Coût total : 9.000€ de chiffre d'affaires perdu.

Leçon: Un playbook Ansible versionné sur Git + des backups S3 chez un autre provider auraient permis de remonter en quelques heures. Il a investi 1 journée pour mettre ça en place, et ne s'inquiète plus.

CAS 2 // Consultante DevOps à Bruxelles — parité dev/prod impossible

Contexte: Amira travaille en freelance pour 3 clients. Chacun a un setup différent, configuré au fil du temps. Quand un bug survient chez un client, elle ne peut pas le reproduire localement.

Impact: Des heures de debug à l'aveugle. Un bug critique en prod prend 6 heures au lieu de 30 minutes. Le client perd confiance.

Leçon: Docker Compose pour le dev + Terraform/Ansible pour la prod = même config, même résultat. Le bug aurait été corrigé en local avant d'arriver en production.

CAS 3 // Agence web à Charleroi — 15 serveurs, 0 cohérence

Contexte: Thomas a une micro-agence (3 personnes). Chaque serveur client configuré à la main, à des moments différents. Versions de PHP incohérentes, configs nginx disparates, SSL renouvelé manuellement.

Impact: Un certificat SSL expire un samedi : le site e-commerce d'un client affiche "non sécurisé" pendant 48h. Le client perd 12.000€ de ventes. Thomas doit rembourser.

Leçon: Un template Ansible commun + Certbot automatisé aurait empêché ça. Thomas a standardisé ses 15 serveurs en 3 jours. Plus jamais ce problème.

Avant / Après — la différence concrète

Chapitre 5

Ce qui change quand on passe d'une posture artisanale à une posture maîtrisée.

Sans IaC (artisanal)	Avec IaC (industriel)
01 RECONSTRUCTION D'UN SERVEUR	
Un weekend complet à reconfigurer "de mémoire", avec quelques oublis qui ressortent une semaine plus tard.	Un playbook Ansible se déroule en 30 à 60 minutes, sans intervention. Le serveur reconstruit est strictement identique au précédent.
02 ONBOARDING D'UN NOUVEAU CLIENT	
Réinventer la roue à chaque mission : installation manuelle, configuration spécifique, oublis fréquents.	Bibliothèque de playbooks réutilisables. Un nouveau projet client démarre avec une stack opérationnelle en quelques heures.
03 MISES À JOUR DE SÉCURITÉ	
Reportées par manque de temps. Quand on s'y met enfin, on prie pour que rien ne casse.	Pipeline automatisé qui patch les serveurs chaque semaine, avec rollback automatique en cas d'incident.
04 DOCUMENTATION	
Inexistante ou obsolète. Le savoir est dans la tête du freelance, invisible et non transmissible.	Le code Ansible/Terraform EST la documentation : précise, à jour, auditée par Git history.
05 CONTINUITÉ D'ACTIVITÉ	
Si le freelance est indisponible (maladie, vacances, urgence), tout s'arrête côté client.	Un autre freelance ou le client lui-même peut reprendre les opérations grâce au code documenté et versionné.
06 IMAGE PROFESSIONNELLE	
Perçu comme un "bidouilleur" sympathique mais sans process structuré.	Perçu comme un partenaire technique de niveau entreprise. De quoi défendre un tarif plus élevé et viser des missions plus ambitieuses.

Auto-évaluation de maturité

Chapitre 6

Faites le point en 5 minutes : où en êtes-vous vraiment ?

Pour chaque question, cochez la case qui correspond le mieux à votre situation actuelle.

SITUATION	NON	PARTIEL	OUI
1. Toutes mes configurations serveur sont versionnées dans un dépôt Git privé.	☐	☐	☐
2. Je peux reconstruire un serveur client complet à partir d'un playbook Ansible (sans documentation manuelle).	☐	☐	☐
3. Mes secrets (clés API, mots de passe) ne sont jamais en clair dans le code.	☐	☐	☐
4. J'utilise Docker Compose pour mes environnements de développement local.	☐	☐	☐
5. J'ai automatisé le renouvellement de mes certificats SSL (Certbot, Caddy ou équivalent).	☐	☐	☐
6. Mes sauvegardes sont automatisées avec un script + cron + stockage externe.	☐	☐	☐
7. J'ai un pipeline CI/CD basique (lint + tests + déploiement auto sur push).	☐	☐	☐
8. Je teste mes playbooks Ansible sur un serveur vierge au moins une fois par projet.	☐	☐	☐
9. Je documente mes choix techniques dans un README au format markdown dans chaque repo.	☐	☐	☐
10. En cas d'absence prolongée, un autre freelance pourrait reprendre mes opérations grâce au code.	☐	☐	☐

SCORING : Non = 0 pt • Partiel = 1 pt • Oui = 2 pts

< 40% : Critique • 40-70% : À renforcer • > 70% : Bonne posture

Ce qu'on met en place pour vous

Chapitre 7

Des outils de grand compte, adaptés à la réalité d'un indépendant.

Audit IaC Express (1h, en visio)

On regarde ensemble votre setup : serveurs, configs, déploiements, sauvegardes. En 1h, on identifie ce qui est fragile, ce qui n'est pas documenté, et ce qu'on peut automatiser en priorité.

Kit IaC Starter — votre infra en code en 2 jours

On traduit votre infrastructure existante en code : Ansible pour la configuration, Terraform pour le provisioning cloud, Docker Compose pour le dev. Tout versionné, documenté, reproductible. Et on vous forme pour que vous soyez autonome.

Pipeline de déploiement clé en main

Un push sur Git et votre application se déploie automatiquement en production, tests inclus, avec rollback automatique si ça casse. GitLab CI, GitHub Actions ou Gitea : on s'adapte à vos outils.

Maintenance IaC récurrente

Check trimestriel : votre code IaC est-il à jour ? Vos dépendances sont-elles sécurisées ? Vos sauvegardes fonctionnent-elles ? On vérifie pour vous.

Notre façon de travailler

Une méthode en 4 étapes, progressive, pensée pour votre réalité. Pas de slides : des livrables concrets à chaque étape.

1

VOIR CLAIR

AUDIT

On commence par comprendre votre contexte : cartographie de l'existant, entretiens avec les équipes, identification des vrais risques. Pas de diagnostic à distance, pas d'hypothèses. On part de ce qui existe chez vous.

2

PRIORISER ENSEMBLE

PLAN

Qu'est-ce qui est urgent, qu'est-ce qui peut attendre ? On construit le plan avec vous, pas à votre place. Chaque action est pondérée par son impact, son effort et votre budget. Vous gardez la main sur les décisions.

3

STABILISER

ACTION

On corrige, on durcit, on protège. Des actions concrètes et mesurables, livrées par jalons. À chaque étape, vous validez avant qu'on avance. Pas de surprise, pas de dérive budgétaire.

4

TRANSMETTRE

AUTONOMIE

On documente tout ce qu'on fait, on forme vos équipes, on vous laisse les clés. Notre succès se mesure à votre capacité à continuer sans nous. Pas de dépendance, du transfert de compétences.

Feuille de route en phases

Chapitre 8

Un chemin balisé, du diagnostic à la maturité — étape par étape.

1 — SEMAINE 1-2

Bases Git

- Compte GitHub privé
- Versionner configs
- Habitude commit

2 — SEMAINE 3-4

Premier playbook

- Apprendre Ansible
- Codifier 1 serveur
- Test reconstruction

3 — MOIS 2

Automatisation dev

- Docker Compose
- Env reproductibles
- Secrets management

4 — MOIS 3

Pipeline CI/CD

- GitHub Actions
- Lint + tests
- Déploiement auto

5 — CONTINU

Maturation continue

- Refactoring modules
- Bibliothèque réutilisable
- Veille outillage

Conseil méthodo

Chaque phase est un livrable autonome. Vous pouvez vous arrêter à n'importe quelle étape ou ralentir le rythme selon votre budget et votre charge interne. L'important n'est pas la vitesse — c'est de ne pas reculer.

Cadre réglementaire & bénéfices cachés

Chapitre 9

L'IaC répond à des obligations légales que beaucoup d'indépendants ignorent.

RGPD — Documenter vos mesures techniques

Souvent ignoré

Le RGPD exige de démontrer les mesures techniques de protection des données (art. 32). "Mon serveur est configuré correctement" ne suffit pas : il faut pouvoir le prouver.

BÉNÉFICE CACHÉ :

L'IaC EST votre documentation technique. Vos fichiers Ansible/Terraform prouvent exactement comment votre infrastructure est configurée : firewall, chiffrement, accès. En cas de contrôle APD, vous avez une réponse immédiate et vérifiable.

Continuité de service — obligation contractuelle

Sous-estimé

La plupart des contrats freelance incluent un SLA implicite. Si vous ne pouvez pas reconstruire l'environnement d'un client rapidement, vous êtes en défaut contractuel.

BÉNÉFICE CACHÉ :

Avec l'IaC, vous pouvez prouver que votre infra est reproductible en heures, pas en jours. C'est un argument commercial : une garantie de continuité que vos concurrents ne peuvent pas offrir.

Conservation légale — 7 ans d'archivage

Contraignant

La loi belge impose 7 ans de conservation comptable. Votre infrastructure d'hébergement doit être fiable et documentée sur le long terme.

BÉNÉFICE CACHÉ :

En codifiant votre infrastructure, vous garantissez sa reproductibilité même si vous changez de provider dans 5 ans. Vous pouvez reconstruire exactement le même environnement d'archivage, à tout moment.

Checklist détachable — vos actions prioritaires

Imprimez cette page, cochez au fur et à mesure. Les colonnes EFFORT et IMPACT vous aident à séquencer.

		EFFORT	IMPACT	DÉLAI
☐	Créer un compte GitHub privé et y versionner toutes vos configurations actuelles.	●○○○○	●●●●●	Cette semaine
☐	Suivre le tutoriel officiel Ansible "Getting Started" (gratuit, 4h).	●●○○○	●●●●●	15 jours
☐	Codifier votre prochaine installation client en playbook Ansible.	●●●●○	●●●●●	1 mois
☐	Installer Docker Desktop et apprendre Docker Compose sur un projet simple.	●●●○○	●●●●○	1 mois
☐	Mettre en place Ansible Vault pour gérer vos secrets actuels.	●●○○○	●●●●○	2 semaines
☐	Automatiser vos sauvegardes avec rclone + cron + Backblaze B2 (~2€/mois).	●●○○○	●●●●●	15 jours
☐	Automatiser le renouvellement SSL avec Certbot ou Caddy.	●●○○○	●●●●○	1 mois
☐	Mettre en place GitHub Actions pour un pipeline CI/CD basique.	●●●○○	●●●○○	2 mois
☐	Tester un playbook sur un serveur vierge (provisionner un VPS jetable).	●●○○○	●●●●○	1 mois
☐	Documenter vos repos avec un README structuré (objectif, prérequis, usage).	●○○○○	●●●○○	Continu
☐	Construire une bibliothèque de roles Ansible réutilisables entre clients.	●●●●○	●●●●●	6 mois

		EFFORT	IMPACT	DÉLAI
☐	Apprendre Terraform basique (provisioning d'un VPS DigitalOcean).	●●●○○	●●●○○	3 mois

CONSEIL : Commencez par les actions à fort impact ET faible effort (les "quick wins").

Si vous bloquez sur une action, contactez-nous : on vous oriente gratuitement.

Questions fréquentes

Chapitre 10

Les questions que nos clients posent le plus souvent — et nos réponses sans détour.

Q1. Par où commencer si je ne connais ni Ansible ni Terraform ?

Commencez par Ansible : la courbe d'apprentissage est plus douce que Terraform, et il n'y a rien à installer sur les serveurs cibles (tout passe par SSH). Suivez le tutoriel officiel "Getting Started" (gratuit). Puis codifiez votre prochaine installation client : config SSH, paquets système, fichiers de config nginx/apache. En quelques semaines, vous couvrez l'essentiel de vos cas courants.

Q2. Faut-il apprendre Terraform si je n'utilise que des VPS ?

Pas immédiatement. Terraform brille pour provisionner des ressources cloud (AWS, Azure, OVH API). Si vous n'avez que 2-3 VPS chez OVH ou Hetzner, Ansible suffit. Mais dès que vous travaillez avec des clients qui ont des ressources cloud, Terraform devient un investissement rentable.

Q3. Comment gérer les secrets (mots de passe, clés API) ?

Trois approches selon le niveau : (1) Débutant : Ansible Vault (chiffrement intégré à Ansible, simple). (2) Intermédiaire : .env chiffrés via SOPS + age. (3) Avancé : HashiCorp Vault auto-hébergé. Jamais de secrets en clair dans Git : même un repo privé peut fuiter (employé qui part, fork accidentel).

Q4. Mes clients me demandent du "GitOps", c'est quoi exactement ?

GitOps est une méthode où l'état désiré de l'infra est entièrement décrit dans Git. Tout changement passe par un commit + un peer review, puis s'applique automatiquement (avec ArgoCD ou Flux pour Kubernetes). Pour un freelance solo, le "GitOps light" consiste simplement à : versionner ses configs, faire des PRs même seul (force de revue), et automatiser le déploiement.

Q5. Combien de temps pour rentabiliser l'apprentissage IaC ?

Pour un freelance qui réalise plusieurs projets serveur par an, l'amortissement se compte en mois, pas en années. La courbe est classique : le premier mois, vous êtes plus lent (apprentissage). Ensuite, chaque projet similaire réutilise vos playbooks et va nettement plus vite. Sans compter l'argument commercial (setup codifié, reproductible, auditable) et les missions plus ambitieuses qui deviennent accessibles.

Q6. Docker, c'est utile pour un freelance ?

Oui, fortement. Docker Compose résout le "ça marche chez moi" : votre environnement de dev est strictement identique à celui du staging et de la prod. Plus de surprises lors des déploiements. Apprenez Docker Compose avant Kubernetes : Kubernetes est dimensionné pour des équipes, Docker Compose est parfait pour un solo.

Q7. Comment expliquer la valeur de l'IaC à un client non-technique ?

Évitez le jargon. Dites : "Si demain je suis indisponible, ou si vous voulez changer de prestataire, votre infrastructure est entièrement documentée dans un format que tout le monde peut reprendre. Vous ne dépendez pas de moi, vous dépendez d'un code que vous possédez." C'est un argument de continuité d'activité.

Q8. Quel hébergeur cloud pour un freelance qui démarre l'IaC ?

Pour les VPS classiques : Hetzner (Allemagne, excellent rapport qualité-prix), OVH (France/Belgique, bon support FR). Pour le cloud public moderne : DigitalOcean (le plus simple) puis AWS Free Tier (le plus utilisé en entreprise). Évitez de commencer par AWS si vous n'avez pas de cas d'usage justifiant la complexité.

Glossaire

Chapitre 11

Tous les termes techniques de ce guide, expliqués en français clair.

Ansible

Outil d'automatisation Open Source qui pilote des serveurs via SSH avec des fichiers YAML lisibles. Pas d'agent à installer. Idéal pour configurer, déployer et orchestrer des dizaines à des milliers de serveurs de la même manière.

CI/CD

Continuous Integration / Continuous Delivery : pipeline automatisé qui teste, valide et déploie le code dès qu'il est commité. Pour l'infra : un changement Terraform passe par des contrôles de sécurité, un peer review, puis s'applique automatiquement.

Configuration drift

Écart entre la configuration réelle d'un serveur et celle qui est documentée ou codifiée. Apparaît quand quelqu'un fait une modif "à la main" sans la propager dans le code. Cause #1 des problèmes de reproductibilité.

Crossplane

Alternative moderne à Terraform qui utilise Kubernetes comme plan de contrôle. Permet de décrire l'infrastructure comme des ressources Kubernetes, avec réconciliation continue. Pertinent en environnement cloud-native.

Docker / Docker Compose

Docker emballe une application avec ses dépendances dans un conteneur portable. Docker Compose orchestre plusieurs conteneurs (app + base de données + cache) avec un seul fichier YAML. Élimine le "ça marche chez moi".

GitOps

Méthode où l'état désiré de l'infrastructure est entièrement décrit dans Git. Tout changement passe par un commit + un peer review. Outils : ArgoCD, Flux. Procure auditabilité, traçabilité et conformité par construction.

Helm

Gestionnaire de paquets pour Kubernetes : permet d'installer des applications complexes (avec configuration paramétrable) en une commande. L'équivalent d'apt-get pour Kubernetes.

Idempotence

Propriété d'un script à pouvoir être exécuté plusieurs fois sans changer le résultat final. Un playbook Ansible idempotent peut être lancé 100 fois : il n'applique les changements que si nécessaire.

Internal Developer Platform (IDP)

Plateforme interne en self-service qui permet aux développeurs de provisionner leur infra (base de données, environnement de test, déploiement) sans passer par les ops. Outils : Backstage, Port, Humanitec.

Kubernetes (K8s)

Orchestrateur de conteneurs : déploie, scale et redémarre automatiquement les applications conteneurisées. Devenu standard pour les architectures cloud-natives. Complexe à exploiter, donc pas toujours adapté aux petites structures.

Mutable vs Immutable infrastructure

Infrastructure mutable : on modifie les serveurs en place (patch, config). Infrastructure immutable : on remplace les serveurs entiers (rebuild complet à chaque changement). L'immutable réduit drastiquement les écarts de configuration.

OPA / Conftest

Open Policy Agent : moteur de policies en code (langage Rego). Permet d'ajouter des règles de conformité automatiques aux pipelines IaC : "tout bucket S3 doit être chiffré", "aucun port 22 ouvert sur Internet", etc.

Packer

Outil HashiCorp pour fabriquer des images de machines (AMI AWS, ISO Linux, image VMware) reproductibles. La fondation de l'infrastructure immutable : on construit l'image, on la déploie partout.

Platform Engineering

Discipline qui consiste à construire des plateformes internes (IDP) pour servir les développeurs. L'évolution naturelle du DevOps : des équipes spécialisées qui transforment l'infrastructure en produit interne consommable.

Playbook (Ansible)

Fichier YAML qui décrit une suite de tâches Ansible à exécuter sur des serveurs cibles : installer des paquets, configurer des fichiers, démarrer des services. La "recette" déclarative.

Pulumi

Alternative à Terraform qui utilise des langages de programmation classiques (TypeScript, Python, Go) au lieu d'un DSL. Utile quand on veut réutiliser des bibliothèques existantes ou des constructions complexes.

Reproducibility

Capacité à recréer un environnement à l'identique à partir du code. Le test ultime : "Si ce serveur disparaît, combien de temps pour le reconstruire ?". Avec une IaC mature, on parle d'heures, pas de jours.

SBOM (Software Bill of Materials)

Inventaire détaillé de tous les composants logiciels (avec versions) qui constituent une image ou une application. Devenu obligatoire dans plusieurs secteurs régulés. Permet de détecter rapidement les vulnérabilités (ex. Log4j).

Secrets management

Pratique consistant à ne jamais stocker les secrets (clés API, mots de passe, certificats) en clair dans le code. Outils : HashiCorp Vault, AWS Secrets Manager, Azure Key Vault, sealed-secrets pour Kubernetes.

Shift Left

Principe consistant à détecter les problèmes (sécurité, conformité, performance) le plus tôt possible dans le cycle de développement. Pour l'IaC : valider la conformité avant le déploiement, dans le pipeline CI.

Snowflake server

Serveur unique, configuré à la main au fil des années, dont la configuration n'est connue de personne. Si ce serveur disparaît, on n'arrive plus à le reconstruire. À identifier en priorité dans toute démarche IaC.

Terraform

Outil HashiCorp pour décrire l'infrastructure cloud (AWS, Azure, GCP, on-premise) en code déclaratif (langage HCL). Le standard de fait de l'IaC moderne. Maintient un fichier d'état (tfstate) qui décrit la réalité provisionnée.

Terratest / InSpec

Outils de tests pour l'infrastructure : Terratest valide qu'un module Terraform crée bien ce qui est attendu. InSpec teste que des serveurs respectent une politique de sécurité (CIS Benchmarks par exemple).

YAML

Format de fichier lisible par les humains, utilisé par Ansible, Kubernetes, Docker Compose, GitHub Actions. Privilégie la lisibilité, mais reste sensible à l'indentation. Apprivoiser YAML est un prérequis IaC.

Ressources externes

Chapitre 12

Sites officiels, outils gratuits et lectures recommandées pour aller plus loin.

HashiCorp Learn

developer.hashicorp.com/tutorials

Tutoriels gratuits officiels Terraform, Vault, Packer, Consul. Excellents pour démarrer et approfondir. En anglais mais très progressifs.

Ansible Documentation

docs.ansible.com

Documentation officielle Ansible (Red Hat). Très complète, avec exemples réels. Le module Galaxy contient des milliers de rôles communautaires prêts à l'emploi.

Kubernetes Documentation

kubernetes.io/docs

Documentation officielle K8s. Accessible aux débutants avec les Tutorials, et exhaustive pour les experts. Contient les bonnes pratiques sécurité (Pod Security Admission).

CNCF Landscape

landscape.cncf.io

Cartographie de tous les outils cloud-native (Kubernetes, observabilité, sécurité, GitOps). Indispensable pour ne pas se perdre dans l'écosystème.

OWASP DevSecOps Top 10

owasp.org/www-project-devsecops-top-10

Les 10 risques majeurs en DevSecOps avec recommandations actionnables. Référence pour intégrer la sécurité dans les pipelines IaC.

Center for Internet Security (CIS Benchmarks)

cisecurity.org/cis-benchmarks

Benchmarks de configuration sécurisée pour Linux, Windows, Kubernetes, Docker, AWS, Azure, GCP. Gratuits, applicables avec InSpec ou OpenSCAP.

Trivy (scanner sécurité IaC)

trivy.dev

Scanner Open Source d'Aqua Security : vulnérabilités, secrets, misconfigurations, IaC, conteneurs, K8s. Le couteau suisse à intégrer dans tous les pipelines.

Checkov (scanner IaC)

checkov.io

Scanner statique pour Terraform, CloudFormation, Kubernetes, ARM templates. Détecte les misconfigurations et applique les politiques de conformité (NIST, CIS, ISO).

Backstage (IDP open source)

backstage.io

Plateforme open-source créée par Spotify pour bâtir un Internal Developer Platform. Catalogue de services, templates, documentation centralisée : la base d'un Platform Engineering moderne.

FlashModz Enterprise — Documents & guides

flashmodz.be/documents

Notre bibliothèque de guides IaC, automatisation, Linux et cybersécurité pour les structures belges. Gratuit, en français, contextualisé Belgique.

À propos de FlashModz Enterprise

FlashModz Enterprise est un cabinet d'expertise IT opérationnelle, fondé par Jérémy Manet et basé à Farciennes, dans la région de Charleroi. Conseil et mise en œuvre concrète : sécurité, audit et pentest, automatisation Linux/DevOps et formations — pour les particuliers comme pour les entreprises, en Belgique et à l'international. Notre conviction : les ressources, les connaissances pointues et les outils des grandes entreprises doivent être accessibles aux PME et aux indépendants. À leur échelle. À leur budget. Avec la même rigueur.

CE QUE NOUS FAISONS

- Audit sécurité
- Pentest & rapport
- Sprint de stabilisation
- Team Lead fractionnel
- Formations Linux / DevOps

CERTIFICATIONS & RÉFÉRENTIELS

- CWNA · Ruckus
- Cisco Meraki
- Fortinet NSE
- Veeam
- Jamf · Apple
- OWASP / PTES
- NIS2
- ISO 27001
- RGPD

Jérémy Manet

Fondateur · Ingénieur & architecte système/réseau

Plus de 15 ans en architecture et ingénierie système/réseau : infrastructure, Wi-Fi d'entreprise, sécurité et virtualisation. Également management (people & technique) et gouvernance IT en tant que Process Owner COBIT, DORA et NIS2. Certifié CWNA, Cisco Meraki, Fortinet, Veeam, Jamf, Apple. Technologies : Aruba, HPE, Cisco, FortiGate, Nutanix.

Spécialités : Architecture · Réseau · Wi-Fi · Sécurité · Virtualisation · Gouvernance · NIS2 · DORA

De l’expertise IT pour ce qui compte vraiment.

contact@flashmodz.be · +32 471 87 87 07 · flashmodz.be

LinkedIn Facebook TikTok Avis Google

IDENTITÉ LÉGALE

FlashModz Enterprise

DÉNOMINATION	FlashModz Enterprise
FONDATEUR	Jérémy Manet
SIÈGE SOCIAL	Rue du Wainage 18, 6240 Farciennes (Belgique)
BCE / TVA	1035.510.038 · BE 1035.510.038
EU VAT	2.386.268.987
ZONES DESSERVIES	Charleroi et alentours (B2C) · Belgique & international (B2B)
CONTACT	contact@flashmodz.be · +32 471 87 87 07 · flashmodz.be

FlashModz

Automatisez votre infrastructure dès aujourd'hui

Chaque entreprise est unique. Chaque idée a ses contraintes,
son contexte et ses priorités.

C'est pourquoi nous ne proposons pas de grille tarifaire figée.
Nous préférons échanger avec vous, comprendre votre situation
et construire une réponse adaptée à vos vrais besoins.

Parlons de votre situation

EMAIL

contact@flashmodz.be

TELEPHONE

+32 471 87 87 07

WEB

flashmodz.be

RETROUVEZ-NOUS

LinkedIn Facebook TikTok Avis Google

Scannez pour accéder à ce document

BCE 1035.510.038 | Charleroi · Belgique & international