WHERE EXPERTISE MEETS PASSION

Automatisation du réseau cloud : Terraform ne suffit pas

Assises de l'infrastructure – Juin 2026

Agenda

Automatisation du réseau Cloud

Terraform ne suffit pas

OBJECTIVES

IaC & Terraform

- *Concepts et bénéfices*

Erreurs et raccourcis courants

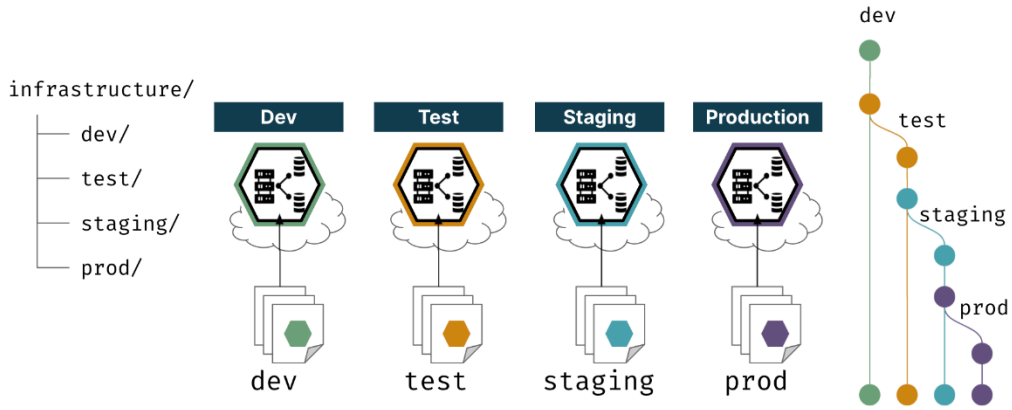
- *5 pièges fréquents et comment les éviter*

Conclusion

- *L'infrastructure gérée comme une application*

Infrastructure as Code

Décrire son infrastructure pour la gérer



Gestion manuelle

Configurations inconsistantes, **non-standard**

Paramétrage en GUI **fastidieux**, procédures

Délais d'implémentation, **erreurs** humaines

Scalabilité impossible à gérer

Infrastructure **décrite, versionnée, reproductible**

Standards et **conception** directement utilisables

Revue, **tests** et **validation** avant déploiement

Automatisation et **cohérence à grande échelle**

Gestion « as code »

Terraform

L'outil d'IaC multicloud par excellence

Langage accessible

- Syntaxe **déclarative et lisible**
- **Adoption** rapide par des profils non-devs

Transversalité

- **Agnostique** des infrastructures
- Théoriquement utilisable pour **toute solution**

Cycle de vie complet

- Création, modification, suppression de **ressources**
- Le code représente la **source d'intention** (« de vérité »)
- Le state reflète l'**état opérationnel** (réel) de l'infrastructure

Couche d'abstraction

- **Gestion transparente** des API et des détails d'implémentation
- Providers et modules **maintenus** par la **communauté** et par les **éditeurs** : AWS, Azure, GCP, Cisco, Palo Alto, F5, etc.

Standard naturel

- [Initialement] open-source et **gratuit**
- **Adoption** massive, large **communauté**
- **Ecosystème** riche (orchestration, tests, etc.)

Erreur #1

Faire faire l'automatisation par « des devs »

Le problème

- Les ingénieurs réseau maîtrisent **le cloud et les protocoles**, mais pas les pratiques de développement
- Les développeurs maîtrisent **l'algorithmie, Git et CI-CD**, mais pas les spécificités réseau
- Résultat : deux **silos**, deux **langages**, des **frictions** permanentes

La solution

- Construire des **squads mixtes** : réseau + infra as code + automatisation
- **Monter en compétence** ensemble : formation croisée, pair programming sur l'infra
- Partager un **langage commun** : code, revues, standards – **dès le départ**

Erreur #2

Fabriquer des pièces uniques avec un processus industriel

Le problème

- Chaque **application** a ses **habitudes**, **contraintes** et **préférences** en matière d'infrastructure
- L'équipe **réseau** a l'habitude de **compenser** et de **s'adapter** à toutes les demandes
- Ingouvernable à l'**échelle**, impossible à **industrialiser**, **dette technique** massive

La solution

- Définir un **catalogue** de ressources et des **designs standardisés** (=> modules Terraform, self-service)
- Obtenir le niveau de **sponsoring/soutien** permettant de **légitimer** ce cadre
- Les **exceptions** deviennent des **demandes d'évolution** – pas des contournements

Raccourci #3

"J'utilise Terraform, donc je suis automatisé"

STRUCTURE DE CODE

- **Structure** : Monolithe, modules, variables, surcouche DRY
- **Règles** : standards, conventions, versioning

SELF-SERVICE MÉTIER

- **Portail** utilisateur : ITSM, custom, etc.
- **Déclenchement** et **suivi** des workflows

PROCESS DE CONTRIBUTION

- **Branches**, **revues**, **approbations**
- Comme pour du code applicatif

ORCHESTRATION

- Plan, **merge**, **apply**, **workflows**
- **Surcouche** : Terragrunt, Terraspace, etc.

OUTILLAGE SPÉCIFIQUE

- Gestion de **config** (Ansible), traitement de **données** (Python), etc.
- **Evènements** (Lambda fonctions), etc.
- **Initialisation**, composants spécifiques

PIPELINES CI-CD

- Analyse de **code** et de **sécurité**
- Exécution de **tests**

Erreur #4

Fermer les yeux sur les mises en production et sur le cycle de vie

Pas d'env. réseau de non-prod

- Tout est développé « sur papier »
- Le debug se fait directement en production

Délégation sans gouvernance

- Plusieurs équipes touchent aux ressources sans règles claires
- Introduction massive de drifts

Absence de tests

- Pas de tests dans le code
- Pas de simulation sur les impacts d'évolution de modules

Remèdes : Gouvernance/RACI, gestion des droits, process de contribution, stratégie multi-environnements, détection proactive des drifts et de revue

Raccourci #5

« L'automatisation, ça se met en place et c'est fini »

L'automatisation, ça s'entretien

Budget
Equipe
Suivi
Communication

Incidents

Amélioration continue

Plateforme

Pipelines

Code

Cas d'usages

Secrets, dépendances

Etc.

Alors comment gérer l'infra « as code » ?

En appliquant les méthodes d'ingénierie logicielle.. en évitant l'usine à gaz

Structure & qualité de code

Organisation, conventions, linting, formatage

Versioning & revue

Git, branches, merge requests, code review

Outillage adapté à l'infra

Potentiellement différents des apps Terragrunt, Terratest, pre-commit,…

Validation & tests

Drifts, tests, policy-as-code, env. de non-production

Gestion en mode produit

Features, roadmap, releases, changelog

CI-CD & pipelines

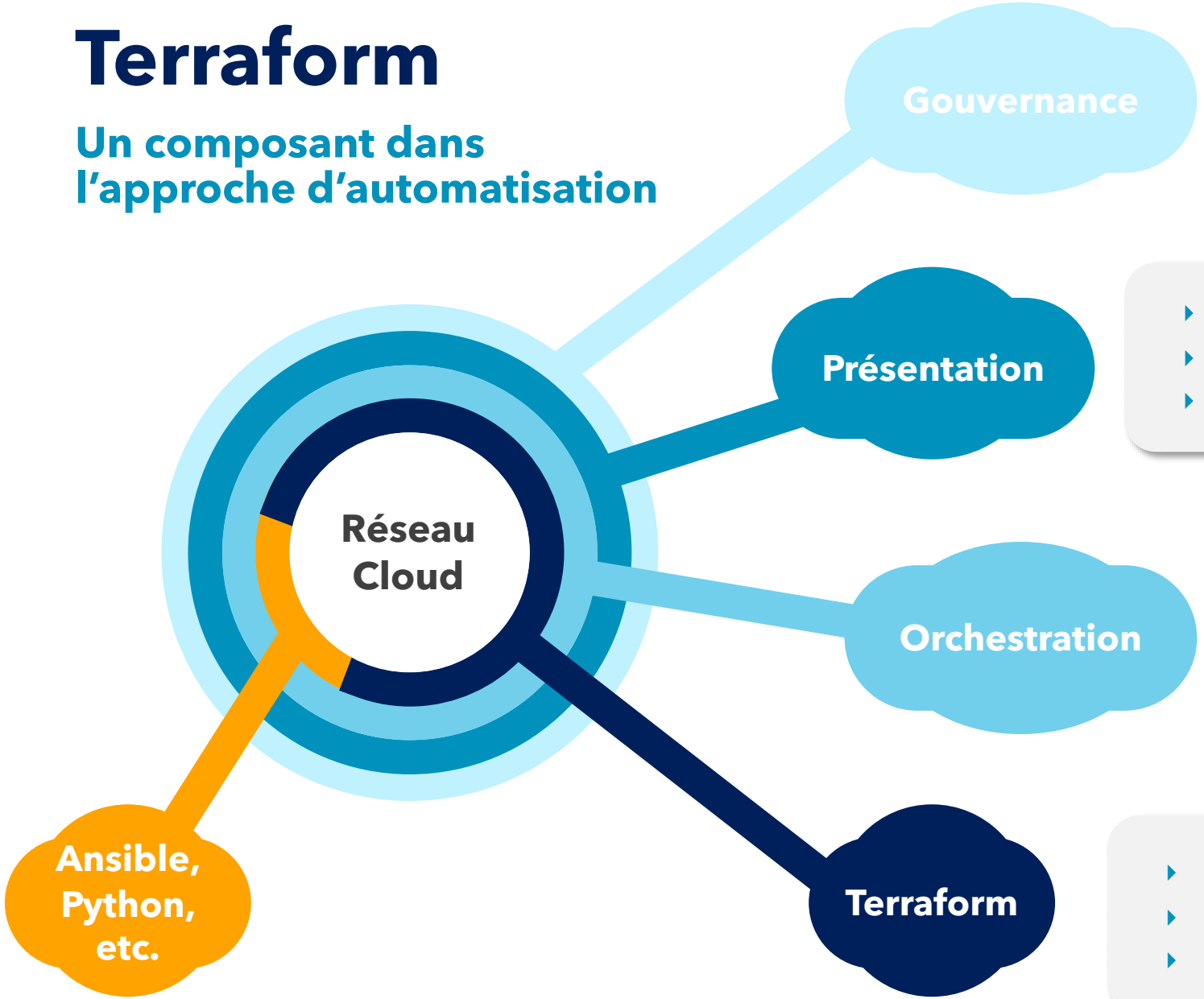
Code, sécurité, tests, plan, apply

Maintien en Conditions Opérationnelles

Code, pipelines, outils : mises à jour, améliorations, incidents, etc.

Terraform

Un composant dans l'approche d'automatisation



- Stratégie, formation, communication, MCO
- Rôles et resp., process de contribution
- Standards & règles de développements

- Catalogue de services
- Self-service et cas d'usages
- Dashboards

- Pipelines et workflows
- Surcouche multi-projets / DRY

- Description de l'intention : code
- Ecriture et maintien des standards : modules
- Déploiement et gestion de l'infrastructure

En résumé

Terraform ne suffit pas

L'automatisation réseau est un système – technique, organisationnel et humain

Provisionnement ≠ Automatisation

Terraform crée des ressources

L'automatisation orchestre, valide, gouverne et opère

Le bon outil pour la bonne tâche

Infra as Code = Software Engineering

Structure, tests, revues, versioning, gestion produit

Les mêmes avantages et exigences que pour une app

Une plateforme + une gouvernance

Plateforme cohérente, modulaire

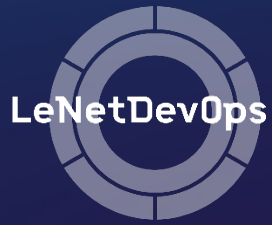
Un catalogue, des règles

Des compétences, une adoption

Du MCO, une communication

Merci

Questions ?



Communauté FR de l'automatisation réseau
https://lenetdevops.fr

Guillaume MAULE
gmaule@cns-com.com
CNS Lyon