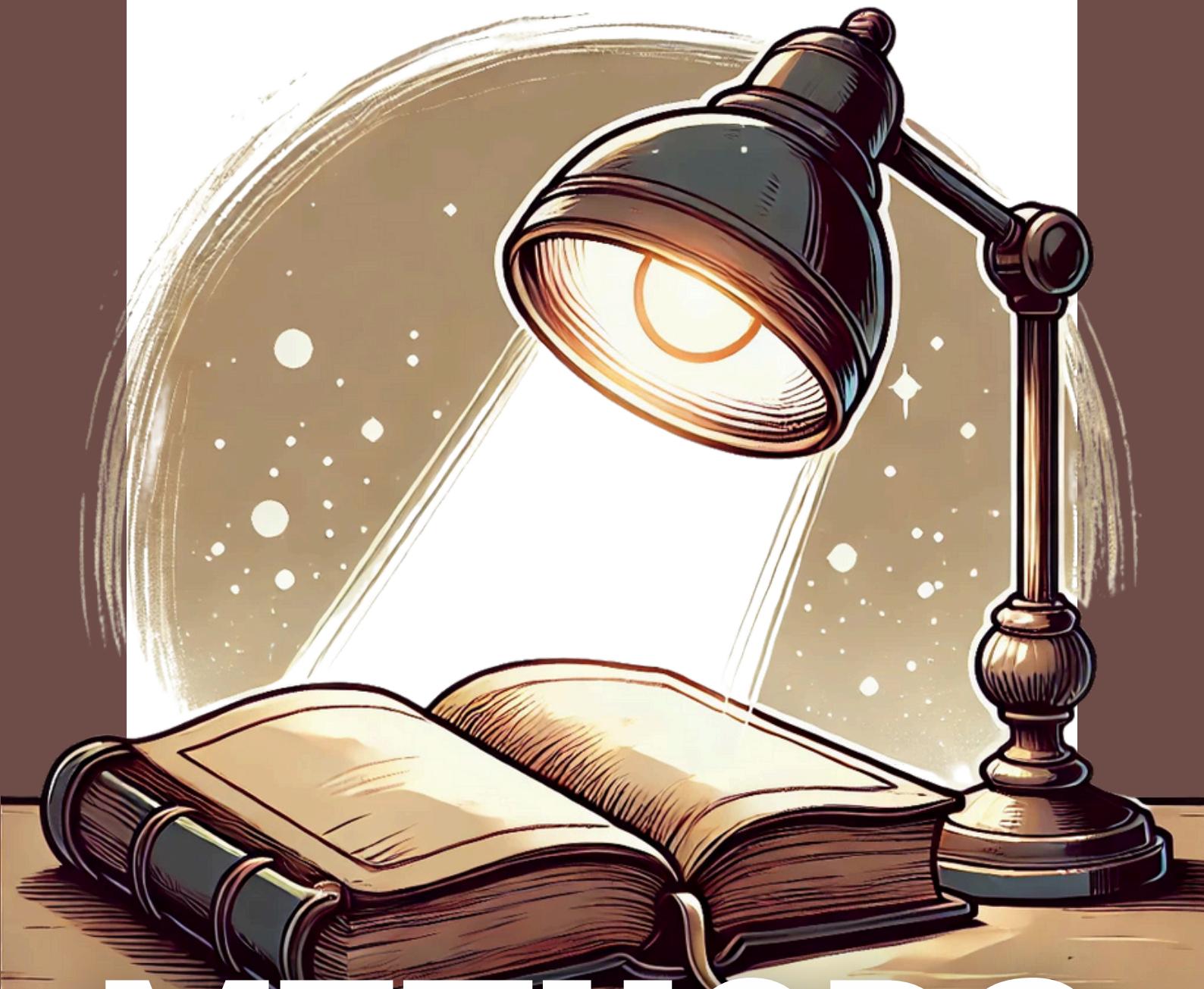


Antony Kervazo-Canut

NAMING CONVENTION

Alignez vos projets



METHODS

SOMMAIRE

Cas d'exemple	3
Nommer ses projets	4
Nommer ses équipes	5
Nommer ses environnements	6
Nommer ses ressources Cloud	7
Établir un document "Naming Convention"	8
Contraintes Techniques et Environnementales	9
Mise en place technique	10
Conclusion	11

Cas d'exemple



Prenons un exemple qui nous servira tout au long de ce document. Cet exemple fictif va nous permettre de nous projeter sur les différentes problématiques abordées (c'est-à-dire) les noms.

Pour cet exemple, prenons la société fictive « Call Me Maybe ». Cette société dispose d'une infrastructure cloud composée de diverses ressources, telles que des machines virtuelles.

Cette société définit un cadre, une convention de nommage pour ses ressources, ses projets, ses documents... bref, tout ce qui porte un nom.



Nommer ses projets



Nommer ses projets est important dans une société, car cela permet de comprendre facilement le but de ceux-ci. Ainsi, on comprend que le projet APICallMe représente une API, là où, s'il s'appelle Utopy, le nom est certes sympa mais n'évoque pas sa fonction.



Si les noms originaux ne sont pas toujours un problème (pourquoi ne pas appeler Sauron un outil de surveillance ?), ils peuvent rapidement le devenir lorsqu'une société renomme des outils tels que Jira ou Confluence en “Orély” ou “Dossymo”, un choix qui risque de perturber les utilisateurs.

Enfin, le nom de votre projet doit être techniquement intelligent. On évite les noms qui pourraient poser problème dans un registre SQL, tels que TABLE, FROM, etc., ou des noms avec des espaces, tels que « Api Call », qui pourrait donner « Api%20Call%20Me » dans une URL ou créer des espaces pénibles dans l'invite de commandes (« Api\ Call\ Me »).

Enfin, essayez d'imaginer que votre projet puisse s'identifier par des initiales. Dans notre exemple, les initiales du projet seraient ACM. Ces initiales permettront ensuite d'avoir des ressources les utilisant plutôt que le nom complet — pratique quand on connaît la limitation du nombre de caractères pour une ressource cloud.

Évitez également d'avoir des noms de projets dont les initiales sont identiques à celles d'autres projets, du moins au sein d'une même équipe.

Nommer ses équipes



Dans une société qui commence à grossir, on a tendance à séparer ses employés en différentes équipes (Front, Back, Middle, etc.). C'est une bonne chose jusqu'à ce que le nombre de projets devienne très important, que la société diversifie ses activités et se retrouve avec plusieurs projets aux fonctions proches.

La première approche revient à créer des noms de projets originaux, mais cela risque de désorienter les ingénieurs travaillant avec plusieurs équipes. Les noms pourraient alors perdre leur sens.

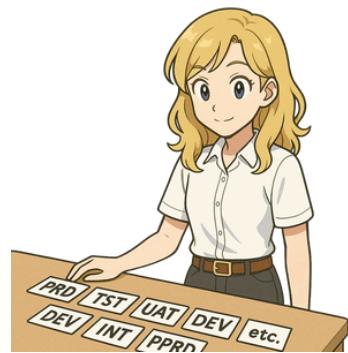
La seconde approche est de nommer les équipes selon la même logique que les projets : un nom clair, accompagné d'initiales. La création d'une ressource cloud pourra alors porter les initiales de l'équipe et du projet. D'un simple coup d'œil, on identifie la fonction de la ressource, pour qui et pour quoi.

Dans notre exemple, la société Call Me Maybe grossit. Elle divise ses ressources humaines en deux équipes : l'équipe Directory, qui va développer les outils de relation client, et l'équipe Marketing, qui va s'occuper de développer le site web de vente de la société.



Attention à ne pas trop cloisonner les équipes. Celles-ci doivent continuer à utiliser des outils communs. Pour faciliter cela, on met en place des ingénieurs capables d'agir sur les différentes équipes (DevOps, DBA, PO, etc.). On évite ainsi d'avoir des équipes qui codent sur GitLab et d'autres sur Azure DevOps : autonomes, mais pas isolées.

Nommer ses environnements



L'environnement est celui dans lequel une application va tourner. Ce n'est pas anodin, et c'est l'endroit où l'on évite absolument les noms "originaux".

Le nom d'un environnement répond à une logique. On n'aurait pas l'idée d'appeler Préprod un environnement qui se déploie après la prod... n'est-ce pas ?

Tous les projets doivent hériter de cette logique commune. Si l'on possède trois environnements : INT, UAT, PRD, tous les projets doivent les utiliser. On évite qu'un projet aille chercher ses données sur INT depuis UAT, ou l'inverse.

On évite également les aléas d'approximation avec des projets référençant « Prod », « Production » ou « PRD » pour le même environnement : restez cohérents.

Les initiales de l'environnement aident aussi à indiquer dans quel environnement se trouve une ressource.



Inutile de se limiter à des environnements à quatre lettres, tels que PPRD. Sur la vie d'une société, un environnement pourrait s'appeler PRD2 dans le cadre d'une migration, ou UAT2 pour réaliser plusieurs tests en parallèles.

Nommer ses ressources Cloud



C'est à partir d'ici que l'ensemble des lignes directrices précédentes prennent tout leur sens. Les noms des ressources vont découler d'une logique descendante :

Nom de l'équipe → Nom du projet → Type de ressource → Environnement.

Dans notre exemple : équipe Marketing, projet APICallMe, environnement de production, type de ressource : Azure Container Apps.

Le nom devient : m-acm-aca-prd.

Pour bien faire, la ressource sera rangée dans un Resource Group (sur Azure) avec un nom similaire : m-acm-rg-prd.



L'ordre des initiales importe peu, l'essentiel est qu'il corresponde aux préférences d'affichage des ingénieurs. Puisque je peux filtrer mes ressources par type, il est par exemple peu nécessaire de commencer le nom par celui-ci.



On évite les actions illogiques, comme numérotter des ressources en 001, 002, 003.... Cela manque de sens : la destruction d'une ressource intermédiaire perturberait la numérotation. Il devient difficile, voire impossible, de prévoir ce nombre avec un outil d'Infrastructure as Code, chaque projet n'embarquant que ses propres ressources.

Établir un document "Naming Convention"



La rédaction d'un tel document doit prendre en compte un maximum d'éléments : le nom des images Docker, des environnements, des projets, des services, des API, des équipes...

Le document doit être accessible à tous afin d'éviter les frustrations et la perte de temps liées à un simple renommage.

Un autre point important est la relecture. Les multiples points de vue permettent d'éviter des erreurs et d'ajouter des informations manquantes.

Enfin, versionnez ce document. Qu'importe la manière dont il est fourni : un numéro de version et une date indiquent aux lecteurs la dernière version. Cela montre aussi que le document n'est pas figé : il évolue et s'adapte à une architecture évolutive.



Contraintes Techniques et Environnementales

Bien sûr, les noms vont faire face à des contraintes auxquelles il faudra s'adapter (sensibilité à la casse, nombre de caractères, etc.). Voici quelques exemples :

AWS S3 Bucket : Longueur entre 3 et 63 caractères.

Azure Resource Group : 90 caractères maximum.

VM Windows Azure : 1 à 15 caractères maximum.

VM Linux Azure : 1 à 60 caractères maximum.

Fichier Windows : Longueur totale du chemin (y compris le dossier) est limitée à 260 caractères.

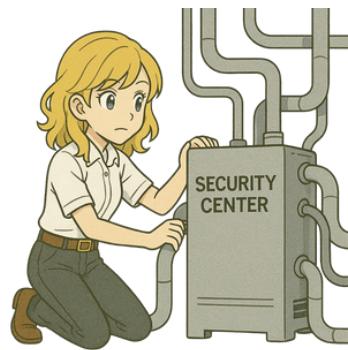
Les plateformes et systèmes imposent également des règles strictes sur les caractères autorisés dans les noms de fichiers, ressources cloud, variables, etc.

AWS S3 Buckets : lowercase, sans espace avec points ou tirets.

SQL : Interdiction d'utiliser comme nom de colonne SELECT, FROM, etc.

Ces exemples montrent qu'il faudra forcément s'adapter pour obtenir des noms cohérents. Mais il faudra tout de même garder du bon sens pour éviter d'inverser l'ordre établit dans le nom des ressources. On reste logique !

Mise en place technique



Définir une convention ne suffit pas. Encore faut-il qu'elle soit respectée. Pour cela, on s'appuie sur des outils capables de contrôler automatiquement les noms utilisés.

Les pipelines d'intégration continue deviennent le premier rempart. Avant même le déploiement, un projet peut échouer si ses ressources ne respectent pas la convention. Les outils comme `tflint` ou `checkov` permettent d'analyser le code Terraform et de rejeter les noms incorrects.

On peut aller plus loin en ajoutant des pre-commit hooks : un développeur ne pourra pas pousser un code qui ne suit pas les règles. La vérification devient immédiate, sans attendre le pipeline.

Les plateformes cloud offrent elles-mêmes des garde-fous. Azure Policy bloque la création d'une ressource si son nom ne suit pas le schéma défini. AWS et GCP proposent des mécanismes équivalents. Ces politiques évitent que des ressources "hors norme" apparaissent par erreur.

Enfin, il est conseillé de centraliser les règles dans un dépôt commun. Ce dépôt devient la référence unique. Chaque évolution est versionnée, ce qui garantit une gouvernance claire et un suivi dans le temps.

Conclusion



Une convention de nommage n'est jamais un simple exercice esthétique. C'est un langage commun qui structure l'ensemble d'une organisation. Elle relie les équipes entre elles, elle guide les machines qui déploient et surveillent les ressources, et elle permet d'éviter les confusions qui coûtent du temps et de l'argent.

Dans notre exemple fictif, la société Call Me Maybe a montré qu'un cadre clair simplifie aussi bien la vie d'un nouvel arrivant que celle d'un ingénieur expérimenté. Les conventions permettent de comprendre en un instant ce qui appartient à quel projet, dans quel environnement, et sous quelle responsabilité.

Mais une convention ne vit que si elle est respectée. C'est pourquoi il est indispensable de l'intégrer dans les outils quotidiens : pipelines, politiques cloud, hooks de validation. Ainsi, elle cesse d'être un document oublié pour devenir une habitude ancrée dans les pratiques.

Enfin, il faut accepter qu'aucune convention n'est définitive. Les entreprises évoluent, les technologies changent, et de nouvelles contraintes apparaissent. Une convention vivante, versionnée et partagée garde toute sa pertinence dans la durée.

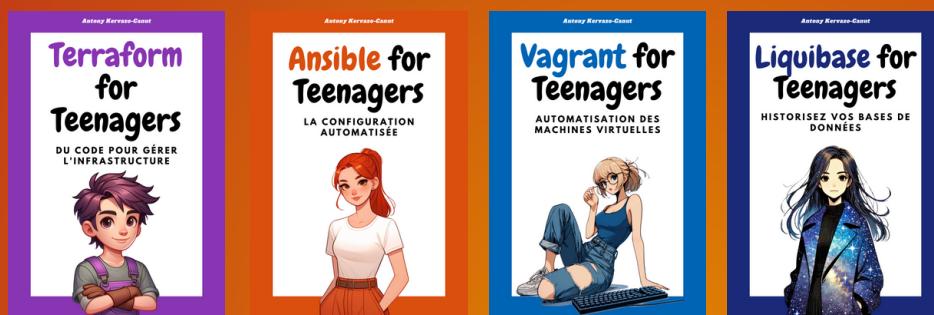
Nommer, c'est gouverner. En donnant un nom clair et logique à ses projets, une organisation se donne les moyens d'être lisible, prévisible et durable.

DevOps Collection

Orchestration et Gestion de Conteneurs



Infrastructure as Code



Sécurité & Gestion des secrets



Forges, Packaging & Cloud



↓ FOLLOW ME ↓



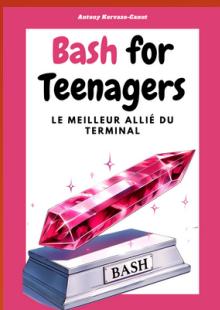
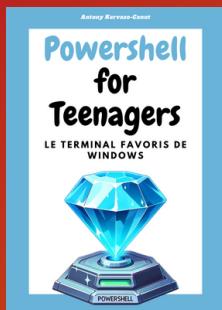
[ANTONYCANUT](#)



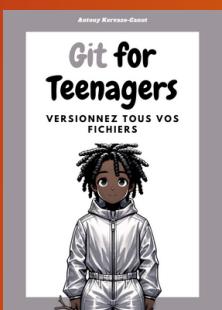
[ANTONY KERVAZO-CANUT](#)

Development Collection

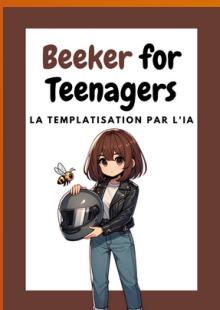
Scripting



Tools



Scaffolding



↓ FOLLOW ME ↓



[ANTONYCANUT](#)



[ANTONY KERVAZO-CANUT](#)

CTO Collection

Organisation



↓ FOLLOW ME ↓



[ANTONYCANUT](#)



[ANTONY KERVAZO-CANUT](#)