21/10/2024

Présentation faite par

TRAN NGUYEN Nori

Variaqua

21 Octobre 2024

Rapport complet du Projet

Présentation faite à

Charles-David WAJNBERG

Remerciements

Je souhaiterais, dans un premier temps, remercier M. Charles-David Wajnberg, responsable de la filière IABD, pour son soutien et son aide tout au long de ce projet. Son encadrement durant ce cursus scolaire m’a été d’une aide précieuse, m’accompagnant dans ma volonté d’obtenir ce *Master Intelligence artificiel et Big Data.*

Je souhaiterais également remercier l’ensemble des intervenants de l’ESGI ayant permis l’acquisition de compétences techniques solides, nécessaires à la bonne réalisation et tenue de ce projet.

Table des matières

1. [Introduction 5](#_Toc180268633)
2. [Sujet 5](#_Toc180268634)
3. [Besoins et objectifs du projet 5](#_Toc180268635)
4. [Contexte 5](#_Toc180268636)
5. [Contrôle de l’existant 5](#_Toc180268637)
6. [Personnes cibles 6](#_Toc180268638)
7. [Objectifs 6](#_Toc180268639)
8. [Gestion du projet 7](#_Toc180268640)
9. [Intervenant 7](#_Toc180268641)
10. [Planification du Projet 7](#_Toc180268642)
11. [*Cahier des charges 7*](#_Toc180268643)
12. [*Diagramme de Gantt 8*](#_Toc180268644)
13. [*Répartition du temps travaillé 8*](#_Toc180268645)
14. [*Répartition des technologies 8*](#_Toc180268646)
15. [Stratégie 8](#_Toc180268647)
16. [Plan du projet 8](#_Toc180268648)
17. [Schéma d’architecture 9](#_Toc180268649)
18. [Les solutions techniques 9](#_Toc180268650)
19. [API Hubeau 9](#_Toc180268651)
20. [Python 10](#_Toc180268652)
21. [*Python 10*](#_Toc180268653)
22. [*Streamlit 10*](#_Toc180268654)
23. [*Modèle LSTM 10*](#_Toc180268655)
24. [Azure 10](#_Toc180268656)
25. [*Blob storage 10*](#_Toc180268657)
26. [*Azure SQL database 11*](#_Toc180268658)
27. [*App service 11*](#_Toc180268659)
28. [Terraform 11](#_Toc180268660)
29. [Github 12](#_Toc180268661)
30. [Utilisation 12](#_Toc180268662)
31. [Présentation de l’application 12](#_Toc180268663)
32. [Recherche d’un code insee 13](#_Toc180268664)
33. [*Documentation utilisateur 13*](#_Toc180268665)
34. [*Documentation technique 13*](#_Toc180268666)
35. [Affichage de l’historique de prédiction 14](#_Toc180268667)
36. [*Documentation utilisateur 14*](#_Toc180268668)
37. [*Documentation technique 14*](#_Toc180268669)
38. [Lancement d’une prédiction 15](#_Toc180268670)
39. [*Documentation utilisateur 15*](#_Toc180268671)
40. [*Documentation technique 15*](#_Toc180268672)
41. [Bilan du projet 16](#_Toc180268673)
42. [*Compétences techniques* 16](#_Toc180268674)
43. [*Difficultés rencontrées* 16](#_Toc180268675)
44. [*Conclusion générale* 17](#_Toc180268676)
45. [Perspective d’évolution 17](#_Toc180268677)
46. [Bibliographie 17](#_Toc180268678)

# Introduction

Dans le cadre de notre diplôme d’expert ingénieur en *Intelligence Artificielle et Big Data* et de notre projet annuel de 5e année, il nous est demandé de réaliser un sujet fonctionnel complet servant un usage clairement identifié et relevant d’une problématique *IABD* de notre choix.

Pour ce faire, nous devons nous appuyer sur plusieurs matières de notre MASTER, que nous développerons plus en détails dans la partie « *Sujet* », ci-dessous.

La pertinence fonctionnelle et technique de ce travail sera, au cours de ce rapport, démontré.

# Sujet

Comme expliqué précédemment, ce projet a donc pour objectif premier de nous faire appliquer les connaissances acquises au sein des multiples matières étudiées en cours.

La contrainte principale est d’inclure, obligatoirement, une matière de type architecture et une matière de type maths/ Machine Learning.

Pour notre projet, nous avons pris le parti d’utiliser différentes matières d’architectures, que nous aborderons et expliquerons au cours de ce rapport, ainsi que du Machine learning.

Cette application doit utiliser un important volume de données et/ ou travailler sur la complexité de leur traitement. Ici, nous nous sommes concentrés sur l’enjeu du volume de ces données.

Celle-ci doit ensuite être déployée dans le Cloud et être auto-alimentée : le dataset doit en effet se compléter automatiquement au fil des prédictions.

Le retour des utilisateurs doit également être pris en compte.

La question des coûts et de leur évaluation doit aussi être abordé.

Nous devons également travailler sur une interface graphique dite « user-friendly » et adapté au public cible.

Ces derniers points vous seront présentés dans les différentes parties à venir.

# Besoins et objectifs du projet

## Contexte

De nos jours, le réchauffement climatique s’intensifie et le niveau des nappes phréatiques n’a jamais autant fluctué entre l’hiver et l’été.

Les sécheresses seront de plus en plus récurrentes. À cause de la hausse des températures, l’évaporation augmentera, renforçant leur intensité et leur durée.

La sécheresse est une menace pour les écosystèmes. En 2050 elle pourrait toucher plus des ¾ de la population mondiale. La pénurie d’eau potable pourrait créer une baisse de la production alimentaire, des conflits pour accéder aux ressources naturelles, ainsi qu’un appauvrissement des populations.

Nous avons pu constater que différentes mesures et actions sont mises en place durant ces périodes de sécheresse afin d’assurer une quantité d’eau par personne suffisante pour la vie quotidienne. Parmi celles-ci, nous pouvons citer une meilleure gestion globale des ressources (par exemple, une communication et sensibilisation aux gestes que nous pouvons accomplir quotidiennement en tant qu’usager), l’optimisation et la restriction des prélèvements d’eau, ou encore l’adaptation des pratiques agricoles aux ressources en eau disponible.

Toutes ces mesures sont, aujourd’hui, destinées à atténuer au mieux ces épisodes climatiques. L’application de celles-ci permet en effet aux nappes phréatiques de conserver un niveau suffisant pendant toute ces périodes, afin de subvenir aux besoins des habitants.

Cependant, aucune des mesures existantes à ce jour ne permet la prédiction, et donc l’anticipation de la durée de ces périodes de stress hydraulique.

Or, prévoir et mieux anticiper la durée de ces épisodes est en effet, à ce jour, la clé pour une meilleure lutte contre ces phénomènes.

## Contrôle de l’existant

Comme nous venons de le soulever, la problématique dans la gestion de ces épisodes, est qu’il n’existe pas, à ce jour, d’application disponible pour le grand public et destinée à prédire le niveau des nappes phréatiques ainsi que leur retour à la normale lorsqu’une zone est en stress hydraulique.

Certaines grandes entreprises comme *Suez* et *Véolia*, possèdent des applications développées en interne mais celles-ci ne sont pas commercialisées et donc inaccessibles aux usagers, premières victimes et acteurs de ces phénomènes.

Pour limiter l'impact de ces évènements climatiques sur les usages de l'eau, tout en préservant les milieux naturels et leur biodiversité, s'adapter est indispensable, d’où l’importance que revêt l’application que nous avons conçu et que nous développerons dans la sous-partie « *Objectifs* » : ***Variaqua***.

## Personnes cibles

Identifier les personnes ciblées par ce projet s’avère être une tâche ardue.

En effet, la France est divisée en 6 bassins hydrographiques, qui n’ont aucun lien avec les circonscriptions administratives françaises. Celles-ci se basent plutôt sur les fleuves français. Ces bassins regroupent toutes les sources d’eau contenues dans une zone géographique (nappes phréatique, château d’eau, …). Un préfet est désigné pour chacun de ces bassins et a autorité pour les décisions à leur échelle.

Cependant, chaque ville, département, région peut choisir d’appliquer des décisions supplémentaires afin de sauvegarder les réserves d’eau.



Dans le cadre de notre application, afin d’obtenir les données les plus riches et précises possibles, il a été choisi de se baser à l’échelle des villes qui possèdent des points de mesure de piézométrie, soit les mesures liées à la pression de la nappe en un point donné.

## Objectifs

Au vu de l’étude du contexte et des besoins préalablement énoncés, *Variaqua* a pour objectif premier de prédire le niveau des nappes phréatiques, la durée de ces épisodes, et le retour à la normale lorsqu’une zone est en stress hydraulique.

A l’issue de ces calculs, l’objectif second et final serait de permettre la régulation de la quantité d’eau utilisée par habitant lors de ces activités climatiques. En effet, après études de ces chiffres, une circonscription pourrait décider d’instaurer la mise en place de mesures et d’actions de restriction et de meilleure gestion globale de l’eau, comme celles que nous avons citées dans l’onglet « *Contexte*. »

Ainsi, il serait possible de notamment réduire la quantité d’eau achetée par une commune pour subvenir aux besoins des habitants sur le long terme.

Avec un prix moyen de 5.02€/m3, une diminution de la consommation permettrait aux villes, surtout celles ne possédant pas de station d’épuration, de faire des économies considérables.

Dans le cadre de ce projet, puisque nous sommes dans le cadre d’un projet scolaire, les moyens dont nous disposons sont limités.

Les ressources utilisées sur notre version actuelle d’*Azure* ne sont donc pas les plus performantes, ce qui ralentit notre programme. Cependant, avec la configuration actuelle, le prix annuel pour l’utilisation d’*Azure*, et donc d’un fonctionnement optimal de notre application, est estimé à 864€/an. Nous avons donc pris le parti d’utiliser notre version de cette plateforme, et de vous avertir de ce point.

# Gestion du projet

## Intervenant

Ce projet a été réalisé individuellement par Nori TRAN NGUYEN, étudiant en 5ème année d’*Intelligence Artificielle et Big Data*.

Les compétences acquises lors de ces 5 années d’études ont permis la bonne tenue de ce projet.

## 

## Planification du Projet

### *Cahier des charges*

Le cahier des charges de ce projet a été envoyé en juillet 2024. Certains critères étaient nécessaires à la validation de cette matière.

Voilà les points importants qu’il nous fallait respecter :

* *Projet spécifique Intelligence Artificielle et Big Data*
* *Application déployée dans le Cloud*
* *Evaluation des coûts*
* *Déploiement automatisé*
* *Prise en compte des utilisateurs*
* *Rapport complet*

Les sujets suivants ne devaient pas être abordés :

* *Pronostic sportif ou trading*
* *Reconnaissance d’image*
* *Système de recommandation basique*

### *Diagramme de Gantt*

Une image contenant texte, ligne, Tracé, nombre

Description générée automatiquementv

### *Répartition du temps travaillé*

Une image contenant texte, capture d’écran, cercle, diagramme

Description générée automatiquement

### *Répartition des technologies*

Ce projet a été principalement composé en *Python.*

Une partie *Terraform* a permis de créer les ressources sur Azure.

Enfin, des fichiers *yml* permettent de déployer le code.

Une image contenant diagramme, capture d’écran, cercle, Graphique

Description générée automatiquement

# Stratégie

## Plan du projet

Le sujet du projet étant très ouvert, la première mission a été de trouver la thématique de celui-ci.

C’est le phénomène caniculaire que nous avons vécu en juillet 2024 qui m’a poussé à m’intéresser au sujet des variations climatiques telles que la sécheresse et qui m’a donc orienté vers la conception de cette application.

Il nous a ensuite fallu rechercher les *API* qui permettraient d’obtenir toutes les données nécessaires à la réalisation du projet.

Une fois ces objectif atteints, une recherche sur le domaine a été effectuée afin de bien identifier les problématiques qui pouvaient être rencontrées ainsi que les diverses informations importantes à faire figurer sur l’application, pour l’utilisateur. Pour répondre à cette question, il nous a fallu, dans un premier temps, bien identifier le public cible afin de fournir les résultats qui lui permettraient de prendre les meilleures décisions pour une gestion optimale de ses ressources.

Dans un cadre plus technique, une fois les données récupérées par le biais de l’*API*, nous avons dû identifier la feuille de route à adopter afin de pouvoir stocker ces informations et les traiter.

Enfin, nous avons travaillé l’interface graphique de manière que l’utilisation soit le plus simple, ergonomique et intuitive possible pour l’utilisateur.

## Une image contenant texte, capture d’écran, diagramme, Police Description générée automatiquementSchéma d’architecture

# Les solutions techniques

## API Hubeau

Afin de récolter nos données, nous avons utilisé l’*API* fournie par *Hubeau* qui recense toutes les informations des stations de mesure depuis des décennies.

Après un traitement des chroniques piézométriques, il a été possible d’isoler des informations essentielles, comme l’identifiant de la station, la hauteur de la nappe phréatique ou encore les dates de mesure.

***Exemple de requête de l’API :***

*https://hubeau.eaufrance.fr/api/v1/niveaux\_nappes/stations?code\_commune=22261&date\_recherche=2024-01-01&format=json&size=1*

## Python

### *Python*

Le langage *Python* a été choisi car il est particulièrement efficace pour l’entraînement de modèles de Machine Learning.

De plus, l’ensemble des librairies misent à disposition permettent de répondre à un grand nombre de besoins. L’affichage graphique par Streamlit, la gestion des tableaux à l’aide de Pandas ou encore l’entrainement de modèles par Tensorflow.

### *Streamlit*

*Streamlit* est une bibliothèque mise à disposition permettant de créer des interfaces graphiques rapidement pour une application *Python*.

Elle a été utilisée dans le cadre de ce projet pour faire l’interface utilisateur.

Le module *folium* a permis de créer la carte interactive offrant la possibilité de choisir une ville.

### *Modèle LSTM*

Le modèle choisi est un modèle *LSTM* (Long Short Time Memory) qui est parfaitement adapté à la prédiction de date.

Principalement utilisé pour le langage, ce modèle a également fait ces preuves dans la prédiction de séries temporelles. Composé de 3 « portes » (*Input, Output et Forget*), *LSTM* est capable de faire le tri entre les informations nécessaires et celles qui sont inutiles.

## Azure

### *Blob storage*

Les *Azure Blob Storage* sont des ressources permettant de stocker des fichiers dans *Azure*. Ils sont ensuite facilement utilisables pour des programmes ou des graphiques.

Ils permettent d’avoir un stockage volumineux, élastique et peu coûteux.

Lorsque le réseau est correctement paramétré, il sera donc possible d’accéder aux données stockées de n’importe où.

Les *storage account* peuvent donc accueillir plusieurs *blob storage*, plus ou moins performants et volumineux en fonction des paramètres choisis.

Les *blob storage* nous ont permis de stocker les *fichiers CSV* extraits de *Hubeau* qui contenaient une liste de ville et d’informations à leur sujet.

### *Azure SQL database*

*Azure* propose un service de bases de données stockées dans le *Cloud*. Elles sont regroupées dans des *Azure SQL Servers.*

Étant facilement utilisables grâce à la librairie *pyodbc* de *Python*, elles proposent toutes les fonctionnalités d’une base de données *SQL Server* et sont facturées à l’utilisation et au volume de stockage, ce qui permet d’adapter efficacement les dépenses liées au stockage.

Les base de données *Azure* nous ont permis de stocker les résultats de nos modèles, ainsi que les coordonnées des villes pour les situer sur la carte géographique.

### *App service*

Les apps service d’*Azure* permettent de déployer des applications sur Internet. En publiant le code sur un *repo Git*, il pourra facilement être mis en lien avec *la web app.*

Il est possible de *build* des applications *Python, Node, PHP, Java*, etc. afin d’obtenir un URL pour y accéder. Comme la majorité des ressources *Azure*, il est possible d’adapter la puissance de l’*App Service Plan* afin de faire tourner même les plus grosses applications.

Notre application a donc été déployée sur une *App Service Azure.*

## Terraform

*Terraform* est une solution proposée par *HashiCorp* permettant de faire du *Platform as Code*.

Cela permet de gérer toutes les ressources d’un *Cloud* efficacement. Particulièrement performant pour gérer les différents environnements de développement, ce langage propose également des modules permettant d’uniformiser le déploiement de ressource entre tous les projets.

Il nous a donc permis de créer toutes les ressources *Azure* citées plus haut et de les paramétrer afin qu’elles soient optimisées pour notre projet.

## Github

Dans un premier temps, *Github* est une plateforme permettant de stocker et partager son code. Chaque développeur peut donc y accéder et modifier le code, tout en gardant un historique.

De plus, *Github* propose des pipelines, à l’aide d’un *fichier yml.*

Il est possible d’exécuter du code sur différents projets ou environnements à l’aide de variable.

Notre code a donc été poussé sur Github et une *pipeline* permet de la *builder* et de la déployer sur notre Web App.

# Utilisation

## Présentation de l’application

Une image contenant texte, carte, capture d’écran, bleu vert

Description générée automatiquement

## Recherche d’un code insee

### *Documentation utilisateur*

Dans l’application, il est possible d’utiliser la carte afin de trouver un code INSEE parmi la sélection de ville proposée.

Pour cela, il suffit de cliquer sur les bulles d’information proposées. Vous obtiendrez ainsi le nom de la ville ainsi que son code INSEE qui vous permettra d’exécuter la suite du programme.

Une image contenant carte, texte, bleu vert

Description générée automatiquement

### *Documentation technique*

Afin de réaliser cette carte, nous avons, dans un premier temps, récupérer un *fichier csv* qui récapitulait une partie des point²s de mesure et leur coordonnées géographiques depuis *Hubeau*.

Ce fichier a été ingéré dans un *blob storage Azure*.

Dans un second temps, nous avons créé une carte sur notre interface utilisateur à l’aide de *folium*.

Enfin, l’application récupère les données stockées dans le *blob storage* afin d’afficher les points sur la carte.

Une description a ensuite été rajoutée lors du clic.

## Affichage de l’historique de prédiction

### *Documentation utilisateur*

L’application permet de consulter les prédictions effectuées précédemment.

Pour cela, il est nécessaire de renseigner le code INSEE dans le champ (n°1) puis d’appuyer sur « afficher l’historique » (n°2).

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquementEnsuite, un tableau s’affichera : vous y trouverez la date de l’entrainement, la ville concernée, le code INSEE, la date du retour à la normale ainsi que le nombre de jours nécessaire pour que la zone ne soit plus considérée comme en stress hydrique.

### *Documentation technique*

Afin de réaliser cette documentation technique, nous avons créé une table dans la base de données *SQL server*. Lorsqu’une prédiction s’achève, les données relevées par cette prédiction sont insérées dans cette table.

Lorsque l’on clique sur « afficher l’historique », une requête est envoyée à la base de données avec une clause *where* en fonction du code INSEE saisi dans le champ précédent. Le résultat de la requête est inséré dans un tableau pandas qui est ensuite affiché et qui permet de consulter les données historisées.

## Lancement d’une prédiction

### *Documentation utilisateur*

Le but principal de l’application étant de prédire un retour à la normale de la hauteur de la nappe phréatique, cette fonctionnalité est très simple.

Une image contenant texte, capture d’écran, Police

Description générée automatiquementPour ceci, renseigner le code INSEE dans le champ de saisie (n°1) puis appuyer sur le bouton « lancement de la prédiction » (n°2).

Après quelques instants de chargement, le résultat sera affiché. Il serait également inséré dans le tableau d’historique et vous pourrez le consulter en utilisant l’affichage de l’historique de prédiction.

### Une image contenant texte, capture d’écran, Police Description générée automatiquement

### *Documentation technique*

Pour cette partie, qui est la plus conséquente du projet, nous avons utilisé *streamlit* pour l’afficher sur l’interface utilisateur.

Lorsqu’un code INSEE est saisi et que le bouton « Lancement de la prédiction » est utilisé, l’application va, dans un premier temps, requêter l’*API* afin de récupérer les relevés des 10 dernières années sur ce site géographique de mesure. Après un rapidement pré-traitement des données, nous utilisons un *dataframe* *pandas* pour entraîner le modèle *LTSM* dessus.

Lorsque la valeur prédite est supérieure à 80% du niveau maximal de la nappe phréatique, nous estimons que le niveau est revenu à la normal. Le nombre de jours est donc calculé. Les résultats trouvés sont insérés dans une table d’une base de données *Azure SQL Server* afin de pouvoir être réutilisée ultérieurement.

# Bilan du projet

## Compétences techniques

Ce projet m’a offert d’utiliser et de tester la bonne acquisition d’une majorité des compétences techniques étudiées en cours.

L’association de ces compétences m’a permis de concevoir une application stable et efficace qui réponds aux attentes fixées à l’origine de mon projet.

Elle répond en effet à la problématique soulevée dans la partie « *Contexte* » de ce rapport, selon laquelle aucune application de ce type n’existe pour le grand public.

Le développement d’une telle application pourrait être réalisé au sein d’une entreprise. Sa création a donc été un exercice très enrichissant.

Enfin, la complexité technique du projet m’a également permis de m’améliorer sur le recueil d’informations en vue de la conception d’une telle application, car il était nécessaire de parfaitement maitriser le sujet abordé.

## Difficultés rencontrées

La principale difficulté de ce sujet fût de me retrouver seul.

En effet, la quantité de travail nécessaire semble impressionnante et il est donc nécessaire de particulièrement bien s’organiser en amont, afin de ne pas être submergé par le nombre de tâches à effectuer.

De plus, il a été nécessaire que je me forme et que j’approfondisse mes compétences à propos de sujets que je ne maîtrisais pas encore, notamment *les bibliothèques frontend de Python* ou encore les *pipelines Guithub.*

Le recueil d’informations a également été une tâche ardue car il y a peu d’informations sur les critères, données et mesure à prendre en compte avant de signaler un état de stress hydraulique.

Enfin, le côté économique a également été un frein dans la réalisation de ce projet car, s’agissant d’un travail scolaire, nous avions des fonds limités et les ressources sur Azure coûtent rapidement cher. Il a donc été nécessaire d’attendre avant de créer les ressources nécessaires et d’optimiser le stockage ainsi que réduire la vitesse de calcul afin de limiter les dépenses.

## Conclusion générale

Pour ce projet, il a été nécessaire d’être inventif afin de trouver un sujet qui serait à la fois novateur, d’utilité publique et d’actualité, tout en respectant les consignes fournies pour le projet.

Ce projet a été le premier, de mon expérience, que j’ai mené de la réflexion théorique (choix du sujet, mise en place du contexte, étude de la cible et des problématiques), en passant par la conception, jusqu’à la mise en œuvre finale.

Ce travail m’a donc permit d’avoir un aperçu du travail que chaque intervenant doit apporter, en entreprise, afin d’aboutir à la bonne réalisation des projets.

Enfin, outre ces aspects techniques et scolaires, la réussite de ce projet pourrait permettre à beaucoup d’utilisateurs de s’organiser efficacement afin de gérer les crises climatologiques.

La bonne utilisation et maitrise de cette application pourrait, à l’issue, permettre d’endiguer la surconsommation inutile d’eau dans les circonscriptions durement atteintes par ces variations climatiques et, à plus long terme, sensibiliser les territoires aux bonnes pratiques de gestion durable de l’eau à l’échelle du territoire.

Ce projet, en plus de m’avoir donné l’occasion d’exploiter mes connaissances déjà acquises, d’enrichir et améliorer mes compétences actuelles, et de me former à de nouvelles pratiques, m’a donc offert la possibilité de travailler sur une application qui s’avèrerait nécessaire à l’avenir pour la bonne utilisation de nos ressources.

J’en ressors donc enrichi.

# Perspective d’évolution

Afin d’améliorer cette application, il serait possible de proposer plusieurs échelles géographiques différentes. Il serait alors possible de prédire au niveau des villes, comme initialement prévu par ce projet, mais également à l’échelle des départements, des régions ou des bassins hydrographiques.

De plus, une possibilité de personnalisation pourrait également améliorer l’application, ce qui permettrait de cibler des zones spécifiques (notamment la « mise en favori » de certaines villes pour une recherche plus efficace.)

Enfin, la réduction du temps de calcul par l’utilisation complète d’Azure pourrait permettre de rendre l’application bien plus fluide.

# Bibliographie

Python 3.10 :

Documentation officielle Python : <https://docs.python.org/3/>

Streamlit 1.38.0 :

Documentation officielle : <https://docs.streamlit.io/>

Forum d’entraide officiel : <https://discuss.streamlit.io/>

Site de documentation : <https://docs.kanaries.net/>

Tensorflow 2.17.0 :

Documentation officielle : <https://www.tensorflow.org/tutorials?hl=fr>

Azure :

Documentation officielle : <https://learn.microsoft.com/en-us/>

Terraform 1.9.8 :

Documentation officielle : <https://developer.hashicorp.com/terraform/docs>

Divers :

Stackoverflow : <https://stackoverflow.co/>

Geeksforgeeks : <https://geeksforgeeks.org>