Rapport sur le déploiement d'une application météo sur Azure Container Instance (ACI) avec GitHub Actions

1. Choix Techniques :

a. Utilisation de Flask pour l'API météo : Flask est un framework web léger en Python qui permet de créer rapidement des API. J'ai choisi Flask pour sa simplicité et sa flexibilité, ce qui en fait un choix idéal pour une petite application comme la mienne.

b. Docker pour la conteneurisation : Docker offre un moyen efficace de packager et de distribuer des applications avec toutes leurs dépendances. J'ai utilisé Docker pour créer une image contenant mon application Flask, ce qui garantit la portabilité et la facilité de déploiement.

c. GitHub Actions pour l'automatisation : GitHub Actions me permet d'automatiser le processus de construction, de publication et de déploiement de mon application. J'ai configuré des workflows GitHub Actions pour déclencher la construction et le déploiement automatiques de mon application à chaque nouveau commit sur la branche principale.

d. Azure Container Registry (ACR) et Azure Container Instance (ACI) : J'ai choisi Azure comme plateforme cloud pour héberger mon conteneur Docker. Azure Container Registry (ACR) est utilisé pour stocker mon image Docker, tandis qu'Azure Container Instance (ACI) est utilisé pour déployer et exécuter mon conteneur Docker.

2. Configurer les Secrets GitHub : Dans les paramètres du repository sur GitHub, configurez les secrets suivants :

AZURE_CREDENTIALS: Informations d'authentification pour Azure.

REGISTRY_LOGIN_SERVER: Lien de la registry Azure Container Registry.

REGISTRY_USERNAME: Nom d'utilisateur pour la registry.

REGISTRY_PASSWORD: Mot de passe pour la registry.

OPENWEATHER_API_KEY: Clé API OpenWeatherMap pour accéder aux données météo.

3. Intérêt de l'utilisation de GitHub Actions :

L'utilisation de GitHub Actions pour automatiser le processus de déploiement présente plusieurs avantages :

a. Automatisation : GitHub Actions permet d'automatiser entièrement le processus de déploiement, ce qui permet d'économiser du temps et d'éliminer les erreurs humaines potentielles associées aux déploiements manuels.

b. Intégration continue : En intégrant le déploiement dans le processus de développement via des actions GitHub, je peux garantir que chaque modification du code est automatiquement déployée et testée, ce qui favorise un cycle de développement rapide et efficace.

c. Traçabilité et reproductibilité : Toutes les étapes du déploiement sont définies dans des fichiers de configuration (comme le fichier YAML du workflow GitHub Actions), ce qui rend le processus de déploiement transparent, traçable et reproductible.

d. Intégration avec d'autres services : GitHub Actions peut être facilement intégré avec d'autres services et outils, ce qui offre une grande flexibilité dans la configuration du processus de déploiement en fonction des besoins spécifiques du projet.