YAML

Qu'est-ce que YAML ?

YAML, qui signifie "YAML Ain't Markup Language" (une récursivité pour "YAML n'est pas un langage de balisage"), est un format de sérialisation de données lisible par l'homme, souvent utilisé pour écrire des fichiers de configuration ou pour échanger des données entre langages de programmation.

Sa simplicité et sa lisibilité ont fait de YAML un choix populaire pour de nombreux outils de développement, déploiement, et automatisation, y compris Docker Compose, les configurations de CI/CD comme GitHub Actions ou GitLab CI, et les systèmes de gestion de configuration comme Ansible.

Caractéristiques

Lisibilité : YAML est conçu pour être facile à lire et à comprendre par les humains, utilisant une indentation pour représenter la structure des données (semblable à Python).

Polyvalence : Il peut représenter les structures de données communes, telles que les scalaires (p. ex., chaînes, nombres), les listes (arrays), et les dictionnaires (objets), ce qui le rend très flexible pour différents cas d'usage.

Compatibilité : YAML est interopérable avec différents langages de programmation, permettant de sérialiser et désérialiser facilement des structures de données complexes.

Utilité de YAML

Fichiers de Configuration : Sa lisibilité et sa simplicité font de YAML un choix idéal pour les fichiers de configuration, où il est important que les configurations soient compréhensibles et modifiables facilement.

Définition d'Infrastructures et de Déploiements : Des outils comme Docker Compose et Kubernetes utilisent YAML pour permettre aux développeurs et aux opérateurs de définir des infrastructures de conteneurs, des réseaux, des volumes, et des politiques de déploiement de manière déclarative.

Automatisation et Orchestration : YAML est largement utilisé dans des systèmes d'automatisation et d'orchestration comme Ansible pour décrire les tâches d'automatisation, les configurations système, et les déploiements d'applications.

CI/CD : Les pipelines d'intégration et de déploiement continus sont souvent configurés via des fichiers YAML, permettant de définir les étapes du pipeline, les environnements, et les scripts d'exécution de manière claire et structurée.

Syntaxe de base de YAML

Voici quelques éléments clés de la syntaxe YAML :

Indentation : YAML utilise l'indentation pour représenter la hiérarchie des données. L'indentation doit être faite avec des espaces, pas avec des tabulations.

Listes : Les éléments d'une liste sont précédés d'un tiret (`-`).

Dictionnaires (Maps) : Les dictionnaires sont des ensembles de paires clé-valeur, où chaque paire est séparée par deux points (`:`).

Commentaires : Les commentaires commencent par un dièse (`#`) et s'étendent jusqu'à la fin de la ligne.

Exemple d'un docker-compose.yml

```
version: '3.8'  # Spécifie la version de la syntaxe Docker Compose utilisée

services:  # Définit les services, c'est-à-dire les conteneurs à exécuter
  web:  # Nom du service web
    image: nginx:latest  # Utilise l'image Docker officielle de Nginx
    ports:
      - "80:80"  # Redirige le port 80 du conteneur vers le port 80 de l'hôte
    volumes:
      - ./html:/usr/share/nginx/html  # Montage d'un volume pour personnaliser le contenu servi par Nginx
    depends_on:
      - db  # Indique que le service web dépend de la base de données et doit attendre son démarrage

  db:  # Nom du service de base de données
    image: postgres:13  # Utilise l'image Docker officielle de PostgreSQL version 13
    volumes:
      - db_data:/var/lib/postgresql/data  # Utilise un volume nommé pour la persistance des données de la base de données
    environment:  # Définit les variables d'environnement pour la configuration de la base de données
      POSTGRES_DB: exampledb  # Nom de la base de données à créer
      POSTGRES_USER: exampleuser  # Nom de l'utilisateur de la base de données
      POSTGRES_PASSWORD: examplepass  # Mot de passe de l'utilisateur de la base de données
```

```
volumes:  # Déclare les volumes utilisés par les services
  db_data:  # Nom du volume pour la persistance des données PostgreSQL
```

Dans cet exemple, deux services sont définis :

1. **Service web (`web`)** :
   - Utilise l'image `nginx:latest` pour créer un conteneur Nginx servant de serveur web.
   - Les requêtes sur le port 80 de l'hôte sont redirigées vers le port 80 du conteneur, permettant d'accéder au serveur web depuis l'extérieur du conteneur.
   - Un volume est monté pour personnaliser le contenu servi par Nginx, permettant d'ajouter ou de modifier les fichiers HTML dans le dossier `./html` de l'hôte.
2. **Service de base de données (`db`)** :
   - Utilise l'image `postgres:13` pour créer un conteneur PostgreSQL servant de base de données.
   - Les données de la base de données sont stockées dans un volume nommé `db_data`, assurant leur persistance indépendamment du cycle de vie du conteneur.
   - Les variables d'environnement sont utilisées pour configurer la base de données, y compris le nom de la base de données, l'utilisateur et le mot de passe.

Ce fichier `docker-compose.yml` montre comment utiliser Docker Compose pour configurer et lier ensemble une application web simple et un service de base de données, illustrant l'utilité des volumes, des ports, des dépendances entre services, et des variables d'environnement dans la configuration de services Docker.

Revision #1
Created 10 March 2024 19:09:52 by MASSON Romain
Updated 10 March 2024 19:23:06 by MASSON Romain