



Créer un chatbot IA multi-conteneurs avec Docker Compose



Silas Teixeira · [Suivre](#)

6 minutes de lecture · il y a 3 heures



Écouter



Partager

Dans ce guide, je vais déployer et gérer un chatbot IA multi-conteneurs à l'aide de Docker Compose.

L'application est une application multi-conteneurs avec trois services : une interface Web, une API backend et un serveur de modèles.

Le serveur de modèles exécute Ollama pour exécuter une instance locale d'un grand modèle de langage (LLM) pré-entraîné pour répondre aux questions liées aux développeurs et garder les réponses courtes.

COMPOSE - LE TLDR

Nous créons des applications cloud natives modernes en écrivant de petits services spécialisés et en les combinant pour créer une application pertinente. Nous les appelons applications microservices, et elles offrent de nombreux avantages, tels que l'auto-réparation, la mise à l'échelle automatique et les mises à jour progressives. Cependant, elles peuvent s'avérer complexes.

Par exemple, vous pourriez avoir une application de microservices avec les sept services suivants :

- Interface Web
- Commande
- Catalogue

- Magasin de données principal

- Enregist

Pour le bon fonctionnement de Medium , nous enregistrons les données des utilisateurs. En utilisant Medium , vous acceptez notre [Politique de confidentialité](#) , y compris notre Politique relative aux cookies.

- Authent

- Autorisation

Au lieu de bricoler ces services avec des scripts complexes et de longues commandes Docker, Compose vous permet de les décrire dans un seul fichier YAML appelé fichier Compose. Vous pouvez ensuite déployer l'application entière en exécutant une commande **docker compose up** qui envoie le fichier Compose à Docker, qui se charge du reste.

L'application Chatbot IA

Nous allons déployer l'application de chatbot IA multi-conteneurs illustrée dans la figure 1. Il s'agit d'un clone d'une application Docker officielle disponible sur [Docker Hub](#) .

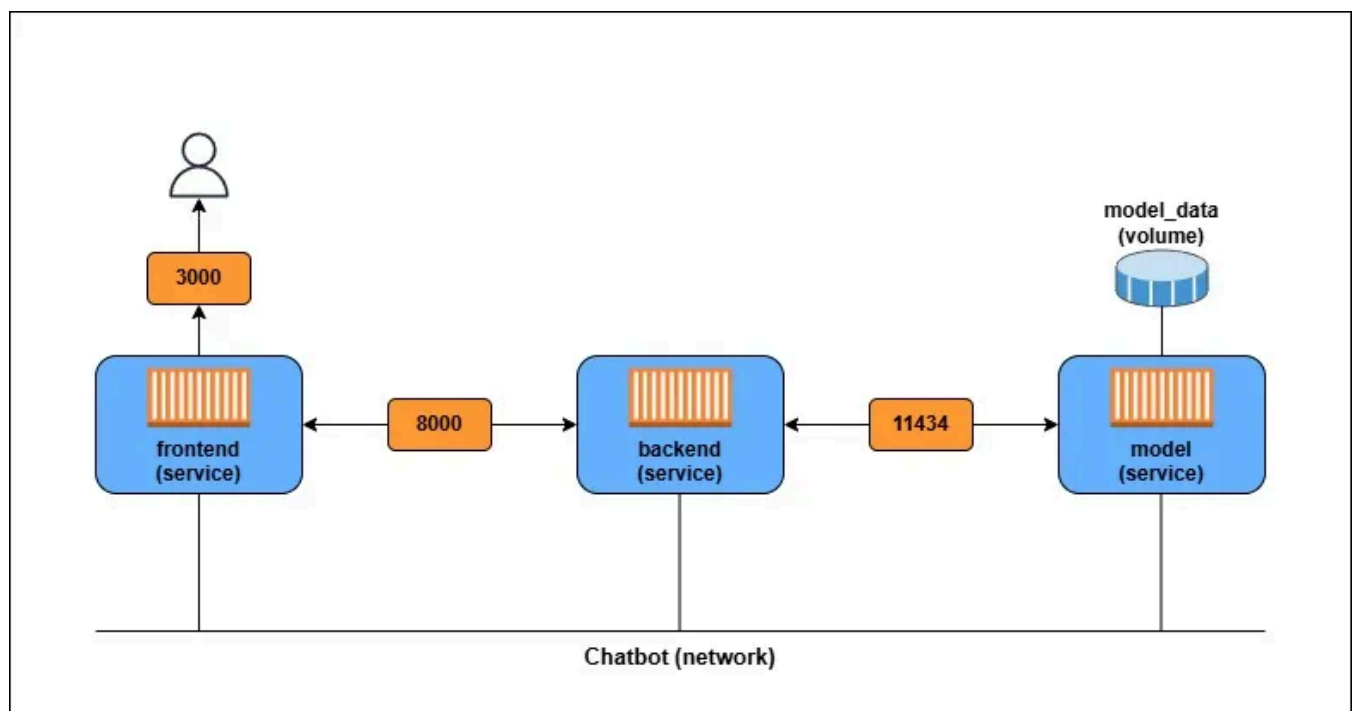


Figure 1 — Diagram

L'application dispose d'un réseau, d'un volume et de trois services. Chaque service exécute une tâche spécifique et, comme illustré dans le schéma, est déployé dans son propre conteneur.

Le service **frontal** exécute un serveur Web qui expose l'interface de chat sur le port 3000 et con

Pour le bon fonctionnement de Medium , nous enregistrons les données des utilisateurs. En utilisant Medium , vous acceptez notre [Politique de confidentialité](#) , y compris notre Politique relative aux cookies.

Le service **model** et transmet les invites du service frontend au serveur modèle sur le port 11434 et diffuse les réponses.

Le serveur **de modèles** utilise l'environnement d'exécution Ollama pour exécuter une instance locale d'un LLM d'instructions Mistral optimisé comme assistant de codage.

Le service de modèle monte le volume **model_data** pour stocker le LLM.

Les trois services se connectent au réseau **de chatbot** .

Tout cela est défini dans le fichier **compose.yaml** du dépôt GitHub suivant : <https://github.com/Silas-cloudspace/multicontainer-ai-chatbot-compose>

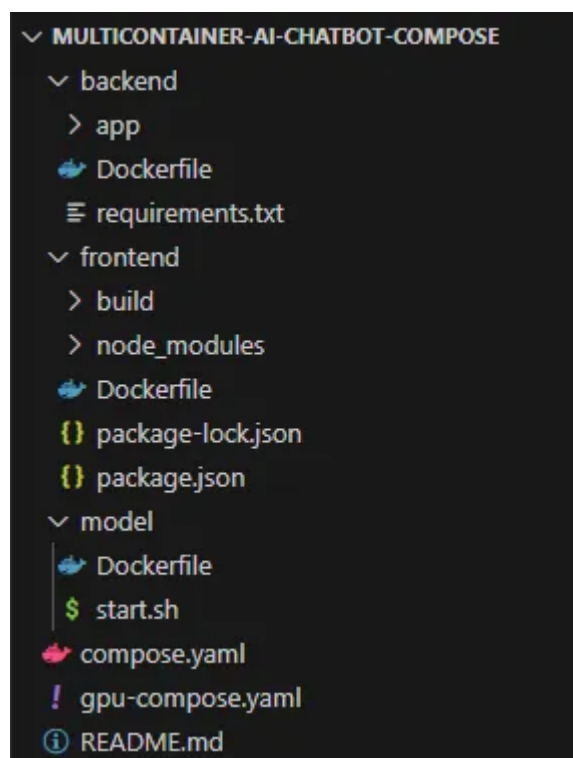


Figure 2 — Répertoires de projets

Le répertoire principal du référentiel contient tout le code de l'application et les fichiers de configuration dont vous avez besoin pour déployer et gérer l'application.

- Le dossier **backend** contient le code de l'application et le Dockerfile pour le service **backend** .

- Le dossier **frontend** contient le code de l'application et le Dockerfile pour le service
- Le dossier **backend** contient le code de l'application et le Dockerfile pour le service **modele**.
- Le fichier **compose.yaml** est le fichier Compose qui décrit l'application (trois services, un réseau et un volume).

Pour le bon fonctionnement de Medium , nous enregistrons les données des utilisateurs. En utilisant Medium , vous acceptez notre [Politique de confidentialité](#) , y compris notre Politique relative aux cookies.

Composer un fichier

Compose définit les applications dans des fichiers YAML appelés « **Compose.yaml** » ou « **compose.yml** ». Vous pouvez toutefois utiliser l' option **-f** pour spécifier un nom de fichier différent lors de l'exécution.

Dans mon cas, mon fichier **compose.yaml** est le suivant :

```
volumes :
  model_data :

réseaux :
  interne :

services :
  frontend :
    image : nigelpoulton/ddd-book:ai-fe
    # build : ./frontend
    commande : npm run start
    réseaux :
      - ports internes
      :
        - cible : 3000
        publié : 3000
    environnement :
      - PORT=3000
      - HOST=0.0.0.0
    depends_on :
      - backend

  backend :
    image : nigelpoulton/ddd-book:ai-be
    # build : ./backend
    réseaux :
      - ports internes : - cible : 8000 publié : 8000 environnement : - MODE
```

Pour le bon fonctionnement de Medium , nous enregistrons les données des utilisateurs. En utilisant Medium , vous acceptez notre [Politique de confidentialité](#) , y compris notre Politique relative aux cookies.

Il est important de noter que le fichier comporte trois clés de niveau supérieur avec un bloc de code sous chacune :

- volumes
- réseaux
- services

Le bloc **volumes** définit un seul volume, le bloc **réseaux** définit un seul réseau et le bloc **services** définit trois services.

```
volumes :  
  données_modèles :
```

réseaux :
inter

Pour le bon fonctionnement de Medium , nous enregistrons les données des utilisateurs. En utilisant Medium , vous acceptez notre [Politique de confidentialité](#) , y compris notre Politique relative aux cookies.

Le bloc « **volumes** » définit un volume unique appelé « **model_data** » que Docker créera sur le système de fichiers de l'hôte. Plus tard, le service **de modèle** montera le fichier et l'utilisera pour stocker le LLM.

Le bloc **réseaux** définit un réseau unique appelé **chatbot** auquel les trois services se connecteront.

Ensuite, nous avons les **services** , et c'est ici que nous définissons les microservices d'application. Ce fichier définit trois microservices : **frontend** , **backend** et **model** .

```
frontend :  
  image : nigelpoulton/ddd-book:ai-fe  
# build : ./frontend  
  commande : npm run start  
  réseaux :  
    - ports internes  
    :  
      - cible : 3000  
      publié : 3000  
  environnement :  
    - PORT=3000  
    - HOST=0.0.0.0  
  depends_on :  
    - backend
```

Ici, Compose demandera à Docker de créer un conteneur appelé **frontend** basé sur l'image **nigelpoulton/ddd-book:ai-fe** .

Il exécutera une commande **npm run start** pour démarrer l'application, utilisera quelques variables d'environnement pour lier l'application au port 3000 sur toutes les interfaces, connectera le conteneur au réseau **de chatbot** et créera un mappage afin que le trafic atteignant l'hôte Docker sur le port 3000 atteigne le conteneur sur 3000.

Enfin, il crée une dépendance sur le service **backend** , garantissant que ce service ne démarrera pas tant que le service **backend** ne sera pas en cours d'exécution.

backend

build

Pour le bon fonctionnement de Medium , nous enregistrons les données des utilisateurs. En utilisant Medium , vous acceptez notre [Politique de confidentialité](#) , y compris notre Politique relative aux cookies.

- ports internes : - cible : 8000 publié : 8000 environnement : - MODE

Le service **backend** est similaire. Compose lit ce bloc et démarre un conteneur appelé **backend** , basé sur l' image **nigelpoulton/ddd-book:ai-be** .

Il exécutera la commande **CMD** ou **ENTRY_POINT** de l'image pour démarrer l'application et injectera la variable d'environnement **MODEL_HOST** dans l'application afin qu'elle sache où trouver le serveur de modèles.

Il connectera le conteneur au réseau **de chatbot** et mapperà le port 8000 sur l'hôte Docker à 8000 sur le conteneur.

Enfin, il crée une dépendance exigeant que le service **modèle** passe un contrôle de santé avant que ce service n'accepte les demandes.

modèle :

image : nigelpoulton/ddd-book:ai-model

build : ./model

réseaux :

- ports internes : - publié : 11434 cible : 11434 volumes : - type : v

Pour le bon fonctionnement de Medium , nous enregistrons les données des utilisateurs. En utilisant Medium , vous acceptez notre [Politique de confidentialité](#) , y compris notre Politique relative aux cookies.

Dans le service **de modèle** , Compose demandera à Docker de créer son image à partir du Dockerfile et des fichiers d'application dans le répertoire **./model** .

Ce service ne définit pas de commande. Il démarre donc l'application à l'aide de la commande spécifiée dans la propriété **CMD** ou **ENTRYPOINT** du **Dockerfile** et **utilise la variable d'environnement MODEL** pour déterminer quel LLM servir. Si le LLM n'existe pas, Ollama l'extrait de la bibliothèque Ollama et le stocke dans **/root/.ollama** , qui repose sur le volume **model_data** .

Il attachera le conteneur au réseau **de chatbot** et mapperà le port 11434 sur l'hôte Docker à 11434 sur le conteneur.

Enfin, il allouera 8 Go de RAM au conteneur et ne se marquera pas comme sain tant que la commande de contrôle de santé ne sera pas terminée avec succès.

Selon les paramètres de vérification de l'état, le conteneur attendra 10 secondes après son démarrage avant de tenter la première vérification. Chaque vérification durera cinq secondes et sera répétée jusqu'à 10 minutes ou 50 fois de plus.

Déployer l'application

Exécutons maintenant la commande **docker compose** pour déployer l'application. Le fichier **compose.yaml** est alors envoyé à Docker, qui le lit et crée le **réseau** de chatbots , le volume **model_data** et un conteneur unique pour chaque service. Le volume **model_data** est ensuite monté dans le conteneur **de modèles** et les trois conteneurs sont connectés au réseau **de chatbots** .

Exécutez la commande ci-dessous pour déployer l'application. Cela peut prendre quelques minutes si votre connexion Internet est lente, car l'application récupère l'image Ollama (environ 1,5 Go), puis le modèle Mistral (environ 4 Go).

docker

Pour le bon fonctionnement de Medium , nous enregistrons les données des utilisateurs. En utilisant Medium , vous acceptez notre [Politique de confidentialité](#) , y compris notre Politique relative aux cookies.

Utiliser l'application

L'application est maintenant opérationnelle et nous pouvons commencer à l'utiliser.

Pour cela, nous devons pointer notre navigateur vers notre hôte Docker sur le port 3000 et poser quelques questions au chatbot.

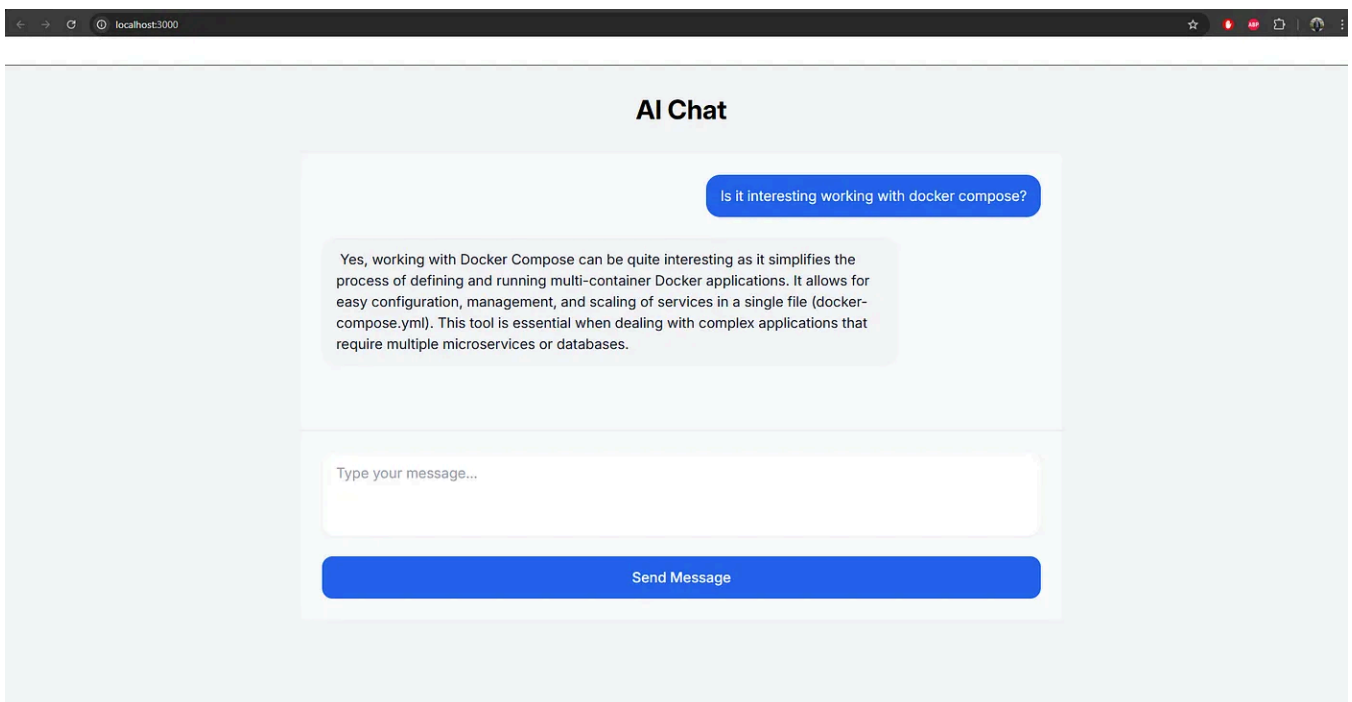


Figure 3 — Chatbot

Nettoyer

Exécutez la commande ci-dessous pour arrêter et supprimer l'application.

```
docker compose down --volumes --rmi all
```

N'utilisez pas les options **--volume** et **--rmi** si vous prévoyez de réutiliser le chatbot prochainement. Cela forcerait Docker à supprimer le volume, le LLM et les images, ce qui rallongerait considérablement le redémarrage de l'application.



Pour le bon fonctionnement de Medium , nous enregistrons les données des utilisateurs. En utilisant Medium , vous acceptez notre [Politique de confidentialité](#) , y compris notre Politique relative aux cookies.

ivre



Écrit par

150 abonnés · 11 Suivant

Aucune réponse pour le moment




Écrire une réponse

Quelles sont vos pensées ?

Plus de Silas Teixeira

No Surveys Available

Be the first to create a survey!

 Create Survey

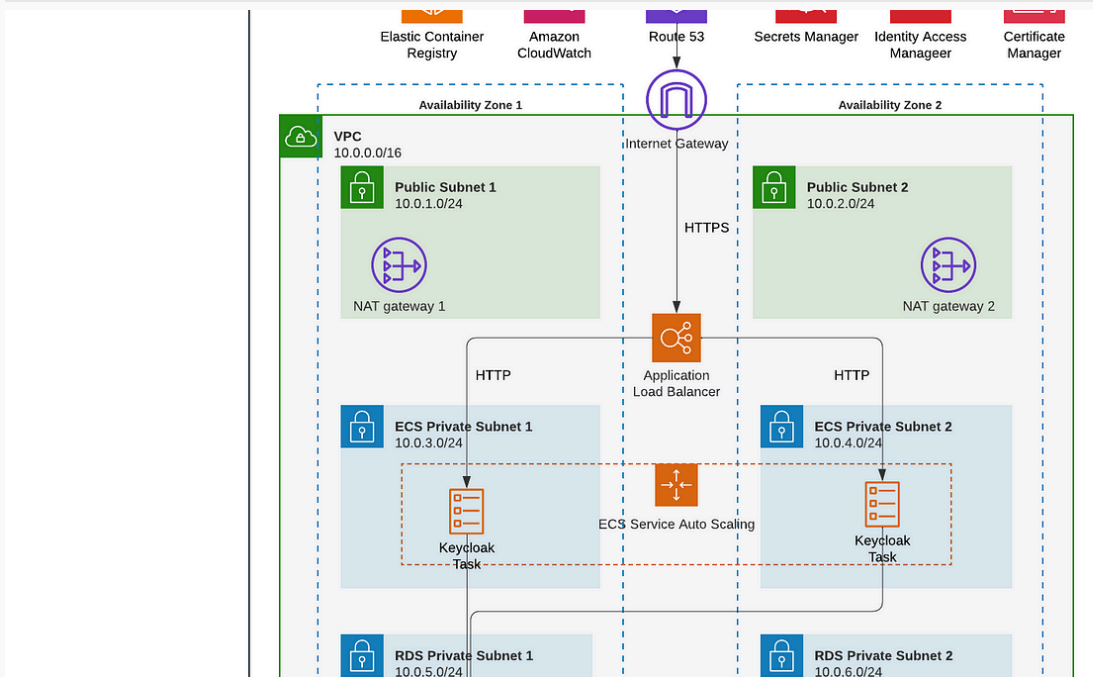
Applicati

Il y a deux se
télécommun

Pour le bon fonctionnement de Medium , nous enregistrons les données des utilisateurs. En utilisant Medium , vous acceptez notre [Politique de confidentialité](#) , y compris notre Politique relative aux cookies.

raform
prises de

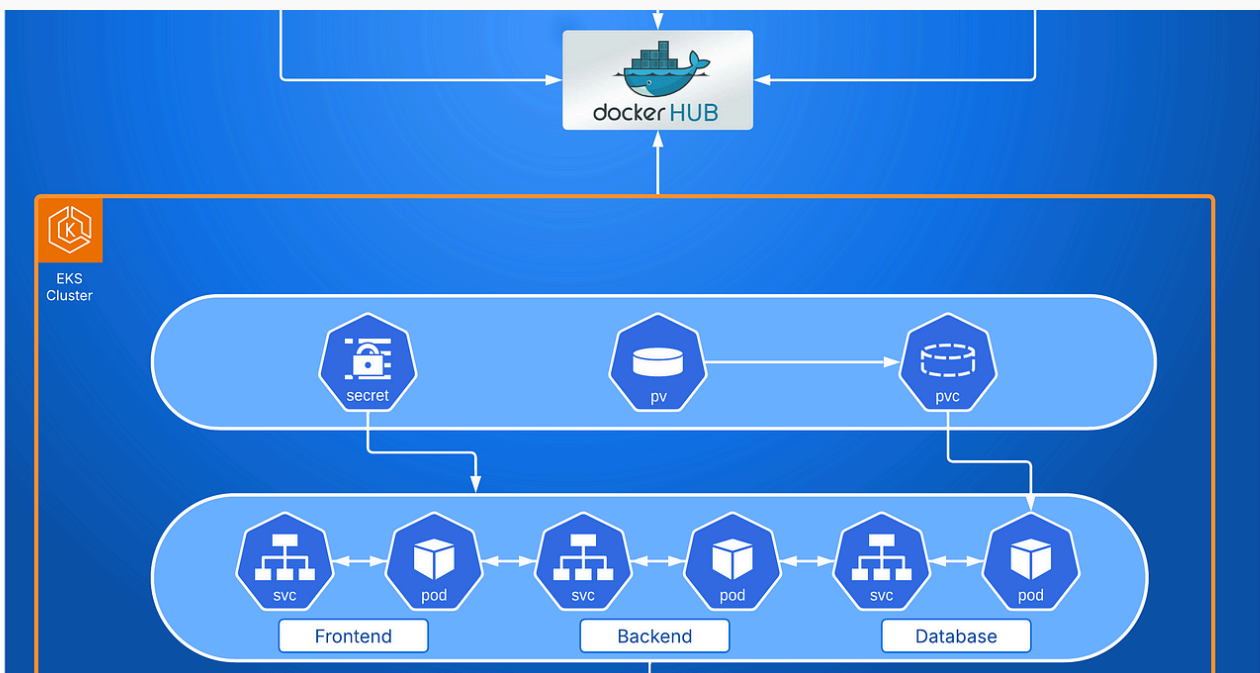
24 mars 2



AWS-CLOUDFORMATION : Déploiement de Keycloak sur ECS avec RDS

GitHub : Découvrez le projet ici.

15 janvier 1 2





Silas Teixeira

Pour le bon fonctionnement de Medium , nous enregistrons les données des utilisateurs. En utilisant Medium , vous acceptez notre [Politique de confidentialité](#) , y compris notre Politique relative aux cookies.

Conteneurs sur AWS EKS

IX sur

GitHub : Découvrez le projet ici.

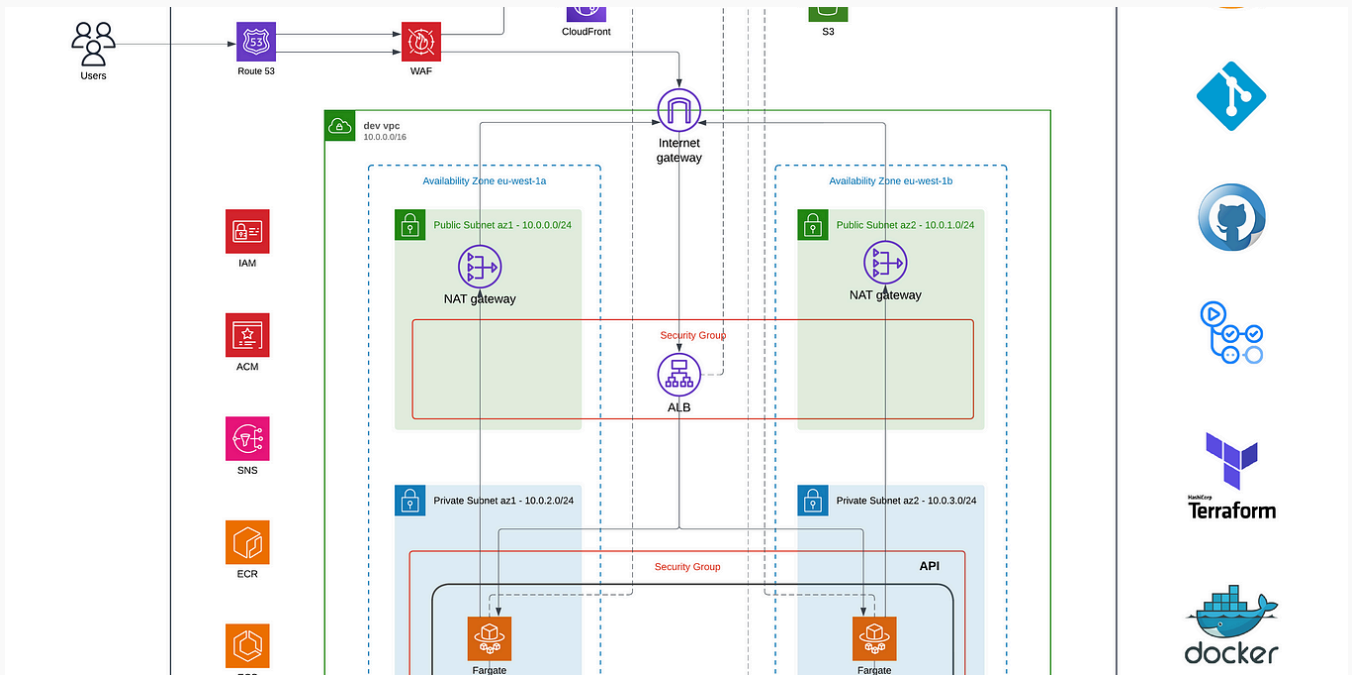
13 mars



5



1



Silas Teixeira

Projets partageant une application Web hébergée sur AWS avec Terraform et GitHub Actions

GitHub : Découvrez le projet ici.

11 février



4



Tout voir de Silas Teixeira

Recommandé par Medium

Pour le bon fonctionnement de Medium , nous enregistrons les données des utilisateurs. En utilisant Medium , vous acceptez notre [Politique de confidentialité](#) , y compris notre Politique relative aux cookies.

RUN AI MODELS LOCALY

Kristiyan Velkov



FE Dans Monde du Front-end par Kristiyan Velkov

Exécutez des modèles d'IA localement : le nouveau AI Model Runner de Docker Desktop

J'ai eu un accès anticipé au nouveau IA Model Runner de Docker Desktop : voici ce que vous devez savoir.

★ 27 mars 🖱️ 229 💬 1



GoPenAI Dans GoPenAI par Ekant Mate (ambassadeur AWS APN)

DevOps optimisé par l'IA sur EKS : comment Amazon Q et CodeWhisperer peuvent accélérer GitOps

Découvrez comment automatiser vos workflows avec les plateformes à automatiser...

✦ il y a 6 jours

Pour le bon fonctionnement de Medium, nous enregistrons les données des utilisateurs. En utilisant Medium, vous acceptez notre [Politique de confidentialité](#), y compris notre Politique relative aux cookies.

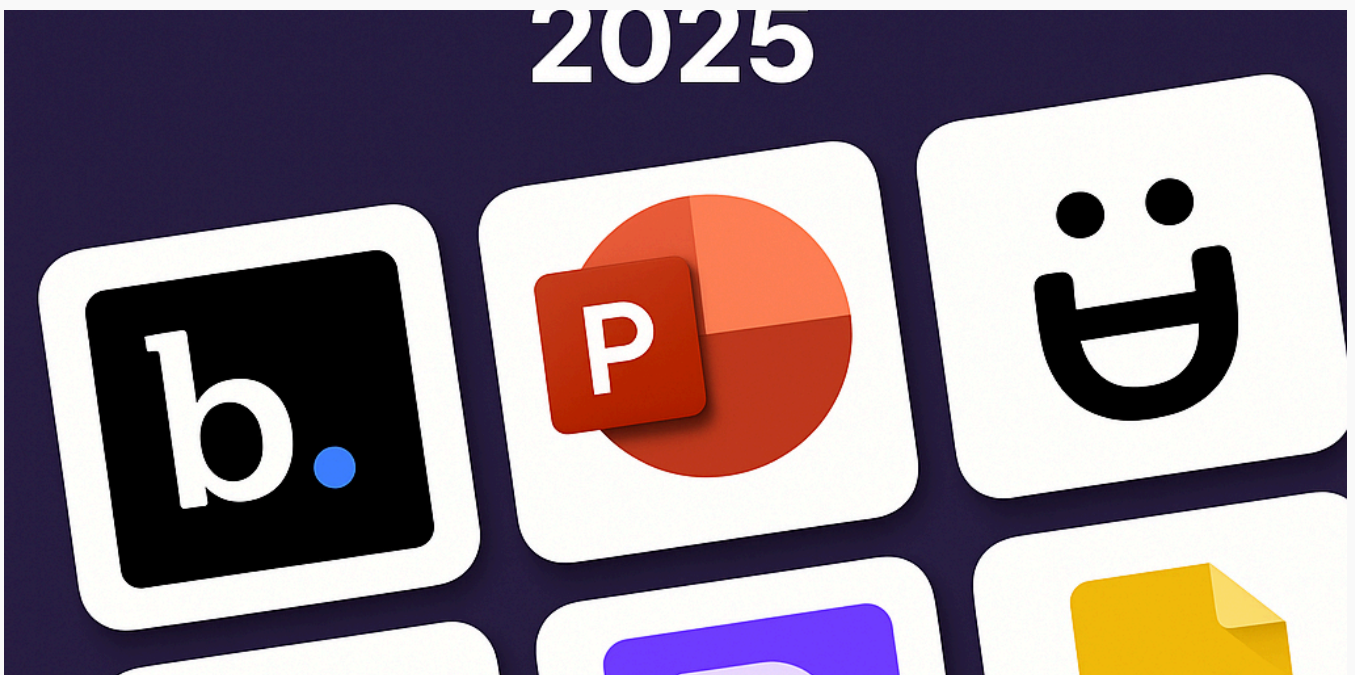


Eren Cankut Uysal

Techniques d'écriture Dockerfile : créer des images efficaces et sécurisées

Créer efficacement des images Docker implique d'optimiser vos Dockerfiles pour la sécurité et les performances. Voici les bonnes pratiques et...

✦ 16 mars 🖱 6





Dans Cod

10 outils de DevOps pour le monde entier

Pour le bon fonctionnement de Medium, nous enregistrons les données des utilisateurs. En utilisant Medium, vous acceptez notre [Politique de confidentialité](#), y compris notre Politique relative aux cookies.

et tout le

L'année où les présentations sont devenues plus intelligentes



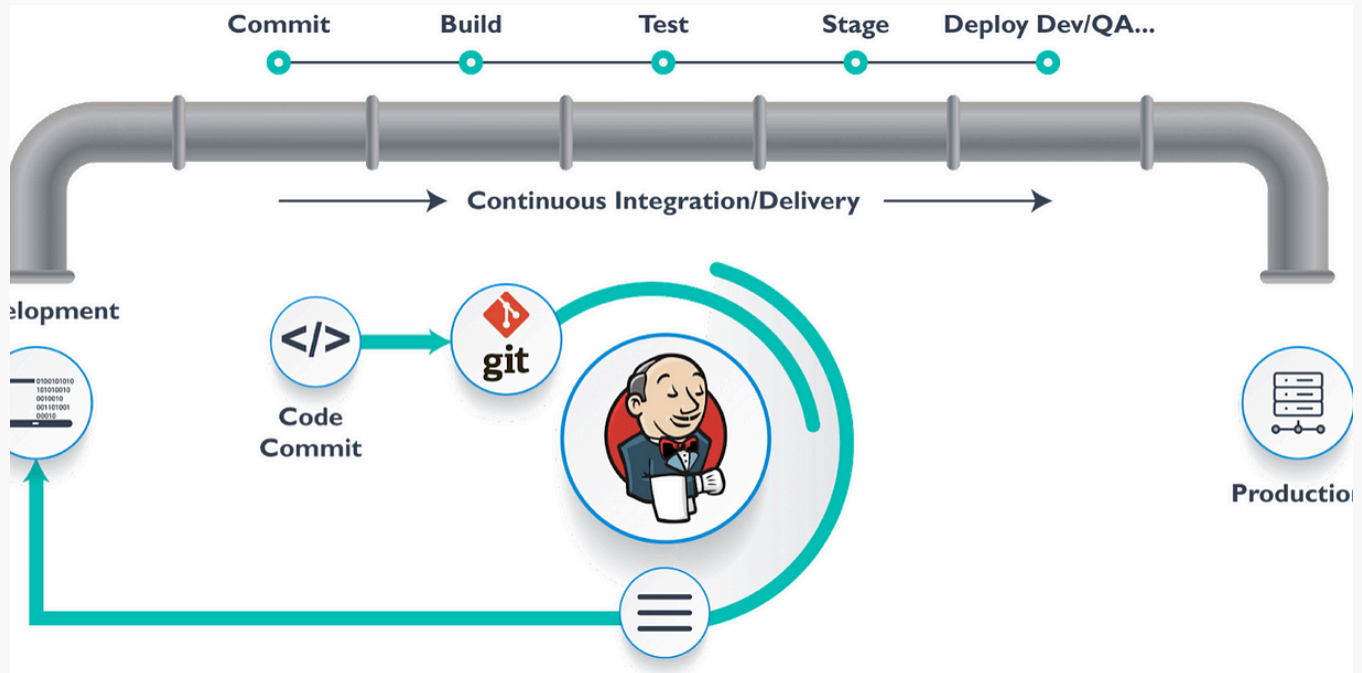
il y a 4 jours



78



1



Armond Holman

Projet DevOps n° 7 : Créer un pipeline CI/CD simplifié

Conteneurisation Docker : un guide complet pour configurer et déployer une application Web via Jenkins Pipeline avec un simple flacon...

24 novembre 2024



4



Pour le bon fonctionnement de Medium , nous enregistrons les données des utilisateurs. En utilisant Medium , vous acceptez notre [Politique de confidentialité](#) , y compris notre Politique relative aux cookies.



How to Set Up and Launch an Amazon EC2 Instance

Cloud For Everybody



Dans Le Cloud pour tous par Aliyan Cheikh

Comment configurer et lancer une instance Amazon EC2

Un guide étape par étape pour configurer, lancer et se connecter à une instance Amazon EC2



24 mars



Voir plus de recommandations