Meilleures pratiques avec Amazon Q Developer pour la génération de code en ligne et par assistant

AWS Conseils prescriptifs

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS Conseils prescriptifs: Meilleures pratiques avec Amazon Q Developer pour la génération de code en ligne et par assistant

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Les marques et la présentation commerciale d’Amazon ne peuvent être utilisées en relation avec un produit ou un service qui n’est pas d’Amazon, d’une manière susceptible de créer une confusion parmi les clients, ou d’une manière qui dénigre ou discrédite Amazon. Toutes les autres marques commerciales qui ne sont pas la propriété d’Amazon appartiennent à leurs propriétaires respectifs, qui peuvent ou non être affiliés ou connectés à Amazon, ou sponsorisés par Amazon.

Table of Contents

Introduction 1
Objectifs 1
Flux de travail des développeurs 3
Conception et planification 4
Codage 4
Révision du code 5
Intégration et déploiement 5
Capacités avancées 6
Transformation du code Amazon Q Developer 6
Personnalisations d'Amazon Q Developer 6
Bonnes pratiques de codage 8
Bonnes pratiques en matière d'intégration 8
Conditions requises pour Amazon Q Developer 8
Bonnes pratiques lors de l'utilisation d'Amazon Q Developer 8
Confidentialité des données et utilisation du contenu dans Amazon Q Developer 9
Bonnes pratiques pour la génération de code 9
Bonnes pratiques en matière de recommandations de code 11
Exemples de code 13
Python exemples 13
Génération de classes et de fonctions 13
Code du document 15
Générer des algorithmes 16
Générer des tests unitaires 19
Java exemples 19
Génération de classes et de fonctions 20
Code du document 23
Générer des algorithmes 25
Générer des tests unitaires 27
Exemples de chat 28
Renseignez-vous sur Services AWS 28
Générer du code 30
Générer des tests unitaires 31
Expliquer le code 33
Résolution des problèmes 37

Génération de code vide 37
Commentaires continus 38
Génération de code en ligne incorrecte 39
Résultats inadéquats des chats 44
FAQs 49
Qu’est-ce qu’Amazon Q Developer ? 49
Comment accéder à Amazon Q Developer ? 49
Quels sont les langages de programmation pris en charge par Amazon Q Developer ? 49
Comment puis-je fournir du contexte à Amazon Q Developer pour une meilleure génération de code ? 49
Que dois-je faire si la génération de code en ligne avec Amazon Q Developer n'est pas précise ? 50
Comment puis-je utiliser la fonctionnalité de chat Amazon Q Developer pour générer du code et résoudre les problèmes ? 50
Quelles sont les meilleures pratiques pour utiliser Amazon Q Developer ? 50
Puis-je personnaliser Amazon Q Developer pour générer des recommandations basées sur mon propre code ? 50
Étapes suivantes 52
Ressources 53
AWS blogues 53
AWS documentation 53
AWS ateliers 53
Collaborateurs 54
Historique de la documentation 55
..... lvi

Bonnes pratiques avec Amazon Q Developer pour la génération de code en ligne et par assistant

Amazon Web Services (Collaborateurs)

août 2024 (1Historique du document)

Traditionnellement, les développeurs s'appuyaient sur leur propre expertise, leur propre documentation et des extraits de code provenant de diverses sources pour écrire et gérer le code. Bien que ces méthodes aient bien servi le secteur, elles peuvent être chronophages et sujettes à des erreurs humaines, ce qui entraîne des inefficiences et des bogues potentiels.

C'est là qu'Amazon Q Developer intervient pour améliorer le parcours du développeur. Amazon Q Developer est un puissant assistant AWS génératif alimenté par l'IA conçu pour accélérer les tâches de développement de code en fournissant une génération de code intelligente et des recommandations.

Cependant, comme pour toute nouvelle technologie, elle peut présenter des défis. Les attentes irréalistes, les difficultés d'intégration, le dépannage des générations de code inexactes et l'utilisation appropriée des fonctionnalités d'Amazon Q sont des obstacles courants auxquels les développeurs peuvent être confrontés. Ce guide complet répond à ces défis en proposant des scénarios réels, des meilleures pratiques détaillées, des solutions de dépannage et des exemples de code pratiquesJava, spécifiques à Python deux des langages de programmation les plus largement adoptés.

Ce guide se concentre sur l'utilisation d'Amazon Q Developer pour effectuer des tâches de développement de code telles que :

- Achèvement du code : générez des suggestions en ligne pendant que les développeurs codent en temps réel.
- Améliorations du code et conseils — Discutez du développement de logiciels, générez du nouveau code en langage naturel et améliorez le code existant.

Objectifs

L'objectif de ce guide est de soutenir les développeurs qui sont des utilisateurs nouveaux ou réguliers d'Amazon Q Developer, en les aidant à utiliser le service avec succès dans leurs tâches de codage

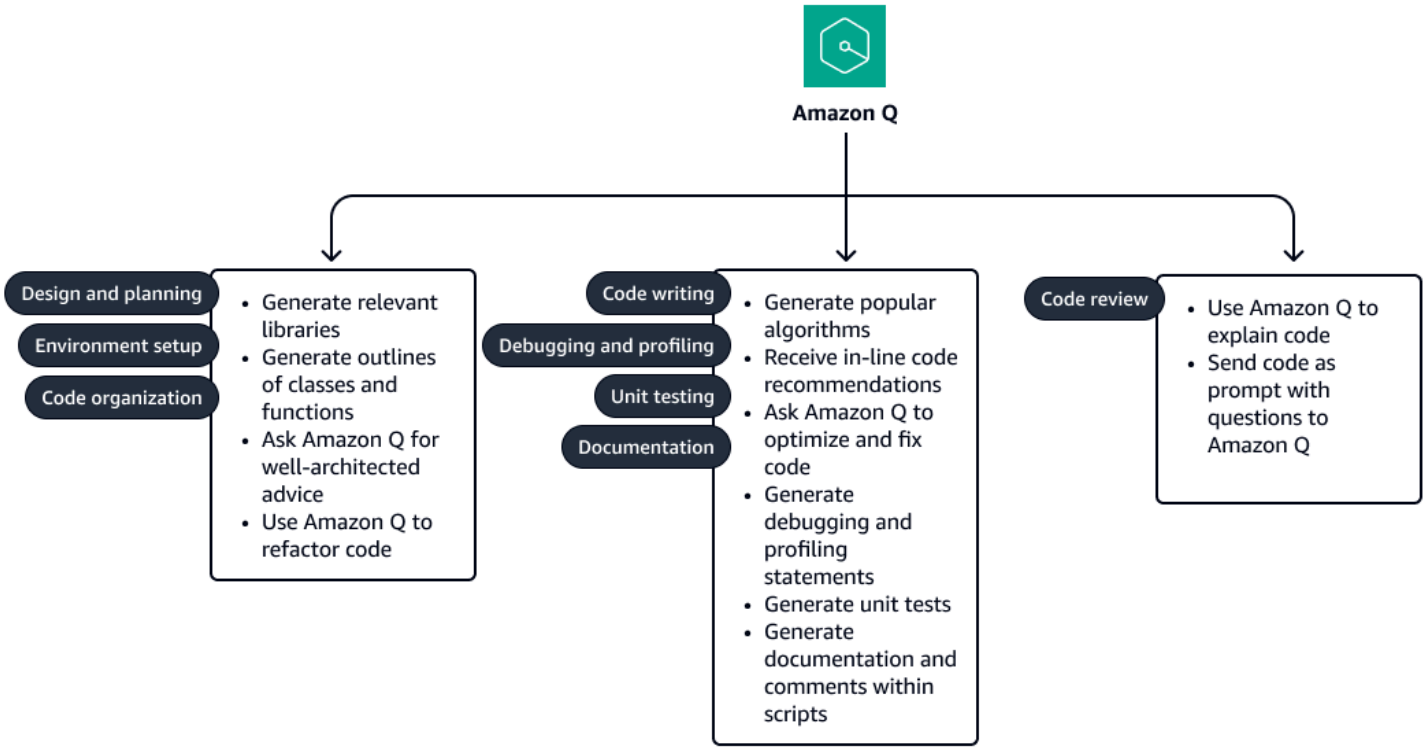
quotidiennes. Les chefs d'équipe de développement peuvent également tirer parti de la lecture de ce guide.

Ce guide fournit les informations suivantes sur l'utilisation d'Amazon Q Developer :

- Découvrez l'utilisation efficace d'Amazon Q Developer pour le développement de code
  - Donnez les meilleures pratiques pour intégrer Amazon Q Developer dans le flux de travail d'un développeur.
  - Offrez step-by-step des conseils avec des exemples pour une génération de code réussie et des recommandations.
- Réduisez les défis courants et encouragez les développeurs à clarifier l'utilisation d'Amazon Q Developer
  - Proposez des stratégies et des informations pour répondre aux attentes des développeurs et surmonter les obstacles liés à la précision et aux performances de la génération de code.
- Fournir des services de dépannage et de gestion des erreurs
  - Fournissez aux développeurs des conseils de résolution des problèmes liés à la génération de code Amazon Q Developer afin de remédier aux résultats inexacts ou aux comportements inattendus.
  - Fournissez des exemples concrets et des scénarios spécifiques à Python etJava.
- Optimisez les flux de travail et la productivité
  - Optimisez les flux de travail de développement de code avec Amazon Q Developer.
  - Discutez des stratégies visant à améliorer la productivité des développeurs.

Utilisation d'Amazon Q Developer dans les flux de travail des développeurs

Les développeurs suivent un flux de travail standard comprenant les étapes de collecte des exigences, de conception et de planification, de codage, de test, de révision du code et de déploiement. Cette section explique comment utiliser les fonctionnalités d'Amazon Q Developer pour optimiser les principales étapes de développement.



Le schéma précédent montre comment Amazon Q Developer peut accélérer et rationaliser les tâches courantes suivantes au cours des étapes de développement du code :

- Conception et planification | Configuration de l'environnement | Organisation du code
  - Générez des bibliothèques pertinentes
  - Génération de plans de classes et de fonctions
  - Demandez à Amazon Q des conseils bien conçus
  - Utiliser Amazon Q pour refactoriser le code
- Écriture de code | Débogage et profilage | Tests unitaires | Documentation
  - Générez des algorithmes populaires

  - Recevez des recommandations de code en ligne
  - Demandez à Amazon Q d'optimiser et de corriger le code
  - Génération d'instructions de débogage et de profilage
  - Générer des tests unitaires
  - Générez de la documentation et des commentaires dans les scripts
- Révision du code
  - Demandez à Amazon Q de vous expliquer le code
  - Envoyez le code avec des questions à Amazon Q

Conception et planification

Après avoir recensé les exigences commerciales et techniques, les développeurs conçoivent de nouvelles bases de code ou étendent les bases de code existantes. Au cours de cette phase, Amazon Q Developer peut aider les développeurs à effectuer les tâches suivantes :

- Générez des bibliothèques pertinentes et des schémas de classes et de fonctions pour obtenir des conseils bien conçus.
- Fournir des conseils pour les questions d'ingénierie, de compatibilité et de conception architecturale.

Codage

Le processus de codage utilise Amazon Q Developer pour accélérer le développement de la manière suivante :

- Configuration de l'environnement - Installez-le AWS Toolkit dans votre environnement de développement intégré (IDE) (par exemple, VS Code ou IntelliJ). Utilisez ensuite Amazon Q pour générer des bibliothèques ou recevoir des suggestions de configuration en fonction des objectifs de votre projet. Pour plus de détails, consultez les meilleures pratiques pour l'intégration d'Amazon Q Developer.
- Organisation du code : refactorisez le code ou obtenez des recommandations d'organisation auprès d'Amazon Q qui correspondent aux objectifs de votre projet.
- Rédaction de code : utilisez des suggestions intégrées pour générer du code lors du développement ou demandez à Amazon Q de générer du code en utilisant le panneau de

discussion Amazon Q de votre IDE. Pour plus de détails, consultez les meilleures pratiques en matière de génération de code avec Amazon Q Developer.

- Débogage et profilage : générez des commandes de profilage ou utilisez les options Amazon Q telles que Fix et Explain pour corriger les problèmes.
- Tests unitaires : fournissez le code sous forme d'invite à Amazon Q lors d'une session de chat et demandez la génération de tests unitaires applicables. Pour plus d'informations, consultez Exemples de code avec Amazon Q Developer.
- Documentation - Utilisez des suggestions en ligne pour créer des commentaires et des docstrings, ou utilisez l'option Explain pour générer des résumés détaillés des sélections de code. Pour plus d'informations, consultez Exemples de code avec Amazon Q Developer.

Révision du code

Les réviseurs doivent comprendre le code de développement avant de le mettre en production. Pour accélérer ce processus, utilisez les options Amazon Q Explain et Optimize, ou envoyez des sélections de code avec des instructions personnalisées à Amazon Q lors d'une session de chat. Pour plus d'informations, consultez la section Exemples de chat.

Intégration et déploiement

Demandez à Amazon Q des conseils sur l'intégration continue, les pipelines de livraison et les meilleures pratiques de déploiement spécifiques à l'architecture de votre projet.

À l'aide de ces recommandations, vous pouvez apprendre à exploiter efficacement les fonctionnalités d'Amazon Q Developer, à optimiser vos flux de travail et à augmenter la productivité tout au long du cycle de développement.

Fonctionnalités avancées d'Amazon Q Developer

Bien que ce guide se concentre sur l'utilisation d'Amazon Q Developer dans le cadre de tâches de programmation pratiques, il est important de connaître les fonctionnalités avancées suivantes :

- Transformation du code Amazon Q Developer
- Personnalisations d'Amazon Q Developer

Transformation du code Amazon Q Developer

L'agent de développement Amazon Q pour la transformation du code peut mettre à niveau la version en langage de code de vos fichiers sans que vous ayez à réécrire le code manuellement. Il fonctionne en analysant vos fichiers de code existants et en les réécrivant automatiquement pour utiliser une version plus récente du langage. Par exemple, Amazon Q transforme un seul module si vous travaillez dans un IDE tel que Eclipse. Si vous utilisez Visual Studio Code, Amazon Q peut transformer l'intégralité d'un projet ou d'un espace de travail.

Utilisez Amazon Q lorsque vous souhaitez effectuer des tâches courantes de mise à niveau du code, telles que les suivantes :

- Mettez à jour le code pour qu'il fonctionne avec la nouvelle syntaxe de la version linguistique.
- Exécutez des tests unitaires pour valider la réussite de la compilation et de l'exécution.
- Vérifiez et résolvez les problèmes de déploiement.

Amazon Q permet aux développeurs d'économiser des jours, voire des mois, de travail fastidieux et répétitif pour mettre à niveau les bases de code.

Depuis juin 2024, Amazon Q Developer prend en charge la mise à niveau Java code et peut transformer Java Code 8 vers des versions plus récentes telles que Java 11 ou 17.

Personnalisations d'Amazon Q Developer

Grâce à ses fonctionnalités de personnalisation, Amazon Q Developer peut fournir des suggestions en ligne basées sur la propre base de code de l'entreprise. La société fournit son référentiel de code soit à Amazon Simple Storage Service (Amazon S3), soit par le AWS CodeConnections biais

de Connections, anciennement connu sous le nom de AWS CodeStar Connections. Amazon Q utilise ensuite le référentiel de code personnalisé doté d'une sécurité activée pour recommander des modèles de codage pertinents pour les développeurs de cette organisation.

Lorsque vous utilisez les personnalisations d'Amazon Q Developer, tenez compte des points suivants :

- Depuis juin 2024, la fonctionnalité Amazon Q Developer Customizations est en mode aperçu. Par conséquent, la disponibilité et le support de cette fonctionnalité peuvent être limités.
- Les suggestions de code personnalisées en ligne ne seront exactes que compte tenu de la qualité des référentiels de code fournis. Nous vous recommandons de consulter un score d'évaluation pour chaque personnalisation que vous créez.
- Pour optimiser les performances, nous vous recommandons d'inclure au moins 20 fichiers de données contenant la langue donnée, tous les fichiers source ayant une taille supérieure à 10 Mo. Assurez-vous que votre référentiel est composé de code source référençable et non de fichiers de métadonnées (par exemple, des fichiers de configuration, des fichiers de propriétés et des fichiers readme).

En utilisant les personnalisations d'Amazon Q Developer, vous pouvez gagner du temps de différentes manières :

- Utilisez des recommandations basées sur le code propriétaire de votre entreprise.
- Améliorez la réutilisabilité des bases de code existantes.
- Créez des modèles reproductibles qui sont généralisés dans l'ensemble de votre entreprise.

Meilleures pratiques de codage avec Amazon Q Developer

Cette section décrit les meilleures pratiques en matière de codage avec Amazon Q Developer. Les meilleures pratiques incluent les catégories suivantes :

- Intégration - Méthodes et considérations à prendre en compte lors de l'intégration
- Génération de code - Conseils pour utiliser correctement la génération de code
- Recommandations relatives au code - Techniques pour améliorer le code

Bonnes pratiques pour l'intégration d'Amazon Q Developer

Amazon Q Developer est un puissant assistant de codage par IA génératif disponible via des logiciels populaires IDEs tels que Visual Studio Code et JetBrains. Cette section se concentre sur les meilleures pratiques pour accéder à Amazon Q Developer et l'intégrer à votre environnement de développement de codage.

Conditions requises pour Amazon Q Developer

Amazon Q Developer est disponible dans le cadre de AWS Toolkit for Visual Studio Code et AWS Toolkit for JetBrains (par exemple, IntelliJ et PyCharm). Pour Visual Studio Code et JetBrains IDEs Amazon Q Developer prend en charge Python, Java JavaScript, C# TypeScript, Go, Rust, PHP, Ruby, Kotlin, C, C++, les scripts Shell, SQL et Scala.

Pour obtenir des instructions détaillées sur l'installation du code AWS Toolkit pour Visual Studio et d'un JetBrains IDE, consultez la section Installation de l'extension ou du plug-in Amazon Q Developer dans votre IDE dans le manuel Amazon Q Developer User Guide.

Bonnes pratiques lors de l'utilisation d'Amazon Q Developer

Les meilleures pratiques générales relatives à l'utilisation d'Amazon Q Developer sont les suivantes :

- Fournissez un contexte pertinent pour obtenir des réponses plus précises, comme les langages de programmation, les frameworks et les outils utilisés. Décomposez les problèmes complexes en composants plus petits.
- Faites des essais et répétez en fonction de vos demandes et questions. La programmation implique souvent d'essayer différentes approches.

- Passez toujours en revue les suggestions de code avant de les accepter, et modifiez-les au besoin pour vous assurer qu'elles correspondent exactement à vos attentes.
- Utilisez la fonctionnalité de personnalisation pour informer Amazon Q Developer de vos bibliothèques internes APIs, de vos meilleures pratiques et de vos modèles architecturaux afin d'obtenir des recommandations plus pertinentes.

Confidentialité des données et utilisation du contenu dans Amazon Q Developer

Lorsque vous décidez d'utiliser Amazon Q Developer, vous devez comprendre comment vos données et votre contenu sont utilisés. Les points essentiels sont les suivants :

- Pour les utilisateurs d'Amazon Q Developer Pro, le contenu de votre code n'est pas utilisé pour améliorer le service ou former des modèles.
- Pour les utilisateurs d'Amazon Q Developer Free Tier, vous pouvez refuser que votre contenu soit utilisé pour améliorer le service via les paramètres ou les AWS Organizations politiques de l'IDE.
- Le contenu transmis est crypté et tout contenu stocké est sécurisé grâce au chiffrement au repos et à des contrôles d'accès. Pour plus d'informations, consultez la section Chiffrement des données dans Amazon Q Developer dans le manuel Amazon Q Developer User Guide.

Bonnes pratiques pour la génération de code avec Amazon Q Developer

Amazon Q Developer fournit des fonctionnalités de génération automatique de code, de saisie automatique et de suggestions de code en langage naturel. Voici les meilleures pratiques pour utiliser l'assistance de codage en ligne d'Amazon Q Developer :

- Fournir un contexte pour améliorer la précision des réponses

  Commencez par le code existant, importez des bibliothèques, créez des classes et des fonctions ou établissez des squelettes de code. Ce contexte permettra d'améliorer de manière significative la qualité de génération du code.
- Codez naturellement

Utilisez la génération de code Amazon Q Developer comme un puissant moteur d'auto-complétion. Codez comme vous le faites habituellement et laissez Amazon Q vous proposer des suggestions au fur et à mesure que vous tapez ou que vous faites une pause. Si la génération de code n'est pas disponible ou si vous êtes bloqué par un problème de code, lancez Amazon Q en tapant Alt+C sur un PC ou Option+C sur macOS. Pour plus d'informations sur les actions courantes que vous pouvez effectuer lors de l'utilisation des suggestions en ligne, consultez la section Utilisation des touches de raccourci dans le manuel Amazon Q Developer User Guide.

- Incluez des bibliothèques d'importation adaptées aux objectifs de votre script

  Incluez les bibliothèques d'importation pertinentes pour aider Amazon Q à comprendre le contexte et à générer du code en conséquence. Vous pouvez également demander à Amazon Q de suggérer des déclarations d'importation pertinentes.

- Maintenir un contexte clair et ciblé

  Concentrez votre script sur des objectifs spécifiques et modularisez les fonctionnalités distinctes dans des scripts distincts avec un contexte pertinent. Évitez tout contexte bruyant ou confus.

- Expérimentez avec des instructions

  Explorez les différentes instructions pour inciter Amazon Q à produire des résultats utiles lors de la génération de code. Essayez, par exemple, les approches suivantes :

  - Utilisez des blocs de commentaires standard pour les instructions en langage naturel.
  - Créez des squelettes avec des commentaires pour renseigner les classes et les fonctions.
  - Soyez précis dans vos instructions, en fournissant des détails plutôt que des généralisations.

- Discutez avec le développeur Amazon Q et demandez de l'aide

  Si Amazon Q Developer ne fournit pas de suggestions précises, discutez avec Amazon Q Developer dans votre IDE. Il peut fournir des extraits de code ou des classes et fonctions complètes pour démarrer votre contexte. Pour plus d'informations, consultez la section Discuter avec un développeur Amazon Q à propos du code dans le guide de l'utilisateur Amazon Q Developer.

Bonnes pratiques en matière de recommandations de code avec Amazon Q Developer

Amazon Q Developer peut répondre aux questions des développeurs et évaluer le code afin de proposer des recommandations allant de la génération de code à la correction de bogues en passant par des conseils en langage naturel. Voici les bonnes pratiques relatives à l'utilisation du chat dans Amazon Q :

- Générer du code à partir de zéro

  Pour les nouveaux projets ou lorsque vous avez besoin d'une fonction générale (par exemple, copier des fichiers depuis Amazon S3), demandez à Amazon Q Developer de générer des exemples de code à l'aide d'instructions en langage naturel. Amazon Q peut fournir des liens connexes vers des ressources publiques à des fins de validation et d'investigation supplémentaires.
- Recherchez des connaissances en matière de codage et des explications sur les erreurs

  En cas de problèmes de codage ou de messages d'erreur, fournissez le bloc de code (avec le message d'erreur, le cas échéant) et votre question sous forme d'invite à Amazon Q Developer. Ce contexte aidera Amazon Q à fournir des réponses précises et pertinentes.
- Améliorez le code existant

  Pour corriger les erreurs connues ou optimiser le code (par exemple, pour réduire la complexité), sélectionnez le bloc de code approprié et envoyez-le à Amazon Q Developer avec votre demande. Soyez précis dans vos instructions pour de meilleurs résultats.
- Expliquer les fonctionnalités du code

  Lorsque vous explorez de nouveaux référentiels de code, sélectionnez un bloc de code ou un script complet et envoyez-le à Amazon Q Developer pour obtenir des explications. Réduisez la taille de la sélection pour des explications plus spécifiques.
- Générer des tests unitaires

  Après avoir envoyé un bloc de code en guise d'invite, demandez à Amazon Q Developer de générer des tests unitaires. Cette approche permet d'économiser du temps et des coûts de développement liés à la couverture du code et DevOps.
- Trouvez des AWS réponses

Amazon Q Developer est une ressource précieuse pour les développeurs qui travaillent avec elle, Services AWS car elle contient une grande quantité de connaissances liées à AWS. Que vous rencontriez des difficultés liées à une activité spécifique Service AWS, que vous rencontriez des messages d' AWS erreur spécifiques ou que vous essayiez d'en apprendre davantage Service AWS, Amazon Q fournit souvent des informations pertinentes et utiles.

Consultez toujours les recommandations qu'Amazon Q Developer vous fournit. Ensuite, apportez les modifications nécessaires et exécutez des tests pour vous assurer que le code répond aux fonctionnalités que vous souhaitez.

Exemples de code avec Amazon Q Developer

Cette section fournit des exemples réalistes de la manière d'améliorer votre expérience et la génération de code en mettant l'accent sur Python and Java langues. Outre des exemples en ligne, certains scénarios montrent comment vous pouvez utiliser la fonctionnalité de chat d'Amazon Q Developer pour demander des recommandations de code pendant que vous programmez.

Pour plus d'informations, consultez les rubriques suivantes :

- Python exemples
- Java exemples
- Exemples de chat

Python exemples

Procédez comme suit : Python des exemples montrent comment vous pouvez commencer à générer des suggestions de code en ligne avec Amazon Q Developer. Ces exemples illustratifs vous montrent comment déclencher l'assistant de codage pour vos propres projets. La plupart des exemples suivants sont applicables à d'autres langues :

- Génération de classes et de fonctions
- Code du document
- Générer des algorithmes
- Générer des tests unitaires

Génération de classes et de fonctions

Pour générer une implémentation de classe complète ou partielle, utilisez des commentaires de code. Le commentaire décrit l'intention de la classe. Commencez simplement et ajoutez plus de détails si nécessaire. Après avoir généré une classe, vous pouvez continuer à générer des fonctions dans son squelette.

Lorsque vous rédigez des commentaires, il est préférable d'utiliser la structure de commentaires de la langue. Entrée Python, pour les commentaires d'une seule ligne, utilisez # .Pour les commentaires multilignes, utilisez ' ' ' ' ' ' ou" " " " " ".

Entrée :

```
'''
DocumentReader class. has a name, children and optional parent.
Includes I/O functions to read and
write document
'''
```

Sortie :

```
class DocumentReader:
    def __init__(self, name, children, parent=None):
        self.name = name
        self.parent = parent
        self.children = children

    def read(self):
        """Reads current document and returns text"""
        pass

    def write(self, text):
        """Writes text to current document"""
        pass
```

Maintenant qu'il existe une classe initiale, une fonction supplémentaire peut être générée avec des commentaires supplémentaires.

Entrée :

```
'''
function to read document from local directory path and print the first line of the
 document
'''
```

Sortie :

```
def read_document(self, path):
        """Reads document from path and prints first line"""
        import os
        if os.path.isfile(path):
            with open(path, 'r') as f:
```

```
                print(f.readline())
        else:
            print(f"File not found at {path}")
```

Code du document

Réduisez le temps consacré à la documentation du code en vous appuyant sur la documentation en ligne. Addition Python pauses dans les commentaires`''' '''`.

Entrée :

```
class S3Utils:
    '''
    <place cursor here>
    '''
    def __init__(self):
        self.base_url = "https://s3"

    def generate_s3_url(self, bucket_name, region):
        return f"{self.base_url}.{region}.amazonaws.com/{bucket_name}"

# Example usage:
if __name__ == "__main__":
    s3_util = S3Utils()
    bucket_name = "my-example-bucket"
    region = "us-west-1"
    s3_url = s3_util.generate_s3_url(bucket_name, region)
    print(f"S3 URL for bucket '{bucket_name}' in region '{region}': {s3_url}")
```

Sortie :

```
class S3Utils:
    '''
    Class to generate S3 URL from bucket name and region
    '''
    def __init__(self):
        self.base_url = "https://s3"

    def generate_s3_url(self, bucket_name, region):
        return f"{self.base_url}.{region}.amazonaws.com/{bucket_name}"

# Example usage:
```

```
if __name__ == "__main__":
    s3_util = S3Utils()
    bucket_name = "my-example-bucket"
    region = "us-west-1"
    s3_url = s3_util.generate_s3_url(bucket_name, region)
    print(f"S3 URL for bucket '{bucket_name}' in region '{region}': {s3_url}")
```

Générer des algorithmes

Amazon Q Developer peut générer des algorithmes populaires, en particulier ceux liés à AWS SDKs. Dans votre script, ajoutez une invite qui indique la fonction que vous souhaitez utiliser.

Cet exemple montre comment générer un dictionnaire de clés d'identification utilisateur pour nommer des valeurs, puis comment générer une fonction de tri à l'aide de l'algorithme de tri à bulles.

Entrée :

```
class SortList:
    @staticmethod
    def get_user_id_name_map():
        user_id_name_map = {
            1: "Amit",
            2: "Rahul",
            3: "Vijay",
            4: "Raj",
            5: "Kumar"
        }
        return user_id_name_map


    '''
    Sort the list of user id's alphabetically using bubble sort
    '''
```

Sortie :

```
    @staticmethod
    def sort_user_id_name_map():
        user_id_name_map = SortList.get_user_id_name_map()
        list_of_user_ids = list(user_id_name_map.keys())
        list_of_user_names = list(user_id_name_map.values())
```

```
        print("Before sort: ", list_of_user_ids)
        for i in range(len(list_of_user_ids)):
            for j in range(len(list_of_user_ids) - i - 1):
                if list_of_user_names[j] > list_of_user_names[j + 1]:
                    list_of_user_ids[j], list_of_user_ids[j + 1] = list_of_user_ids[j +
 1], list_of_user_ids[j]
                    list_of_user_names[j], list_of_user_names[j + 1] =
 list_of_user_names[j + 1], list_of_user_names[j]
        print("After sort: ", list_of_user_ids)
```

En outre, vous pouvez créer des instructions pour les procédures standard. Dans cet exemple, l'utilisateur crée une invite pour stocker le dictionnaire dans une table Amazon DynamoDB. Au début du script, l'utilisateur ajoute une invite à ajouter des bibliothèques DynamoDB afin de fournir un contexte supplémentaire à Amazon Q Developer.

Entrée :

```
'''
Add libraries for Amazon Dynamo DB
'''
```

Sortie :

```
'''
Add libraries for Amazon Dynamo DB and JSON.

iMport necessary libraries for working with Amazon DynamoDB and JSON data.
'''

import boto3
import json
from botocore.exceptions import ClientError

def load_data_to_dynamodb(table_name, data_file):
    # Create a DynamoDB resource
    dynamodb = boto3.resource('dynamodb')

    # Get a reference to the table
    table = dynamodb.Table(table_name)

    # Load data from the JSON file
    with open(data_file, 'r') as file:
```

```
        data = json.load(file)

    # Iterate over the data and load it to the table
    for item in data:
        try:
            table.put_item(Item=item)
            print(f"Successfully loaded data: {item}")
        except ClientError as e:
            print(f"Error loading data: {e.response['Error']['Message']}")

# Example usage
load_data_to_dynamodb('myTable', 'data.json')
```

Désormais, l'utilisateur peut facilement demander à Amazon Q de l'aider à stocker le dictionnaire dans DynamoDB.

Entrée :

```
'''
   function to store the above sorted dict to dynamodb table named users
'''
```

Sortie :

```
@staticmethod
    def store_user_id_name_map_in_dynamodb():
        user_id_name_map = SortList.get_user_id_name_map()
        list_of_user_ids = list(user_id_name_map.keys())
        list_of_user_names = list(user_id_name_map.values())
        dynamodb = boto3.resource('dynamodb')
        table = dynamodb.Table('users')
        for i in range(len(list_of_user_ids)):
            try:
                response = table.put_item(
                    Item={
                        'user_id': list_of_user_ids[i],
                        'user_name': list_of_user_names[i]
                    }
                )
            except ClientError as e:
                print(e.response['Error']['Message'])
```

Générer des tests unitaires

Grâce aux commentaires intégrés, Amazon Q Developer peut générer des tests unitaires pour effectuer des assertions et d'autres tests. Dans cet exemple, l'utilisateur demande à Amazon Q de générer les bibliothèques applicables aux tests unitaires afin de définir le contexte du script de test. L'utilisateur demande ensuite à Amazon Q d'écrire une fonction de test pour cette `sort_user_id_name_map` fonction.

Entrée :

```
'''
Import all the required unit test libraries for python and then generate a unit test
 for sort_user_id_name_map function.
'''
```

Sortie :

```
import unittest
from unittest.mock import patch
from sort_user_id_name_map import sort_user_id_name_map

class TestSortUserIdNameMap(unittest.TestCase):
    @patch('sort_user_id_name_map.sort_user_id_name_map')
    def test_sort_user_id_name_map(self, mock_sort):
        mock_sort.return_value = {'id1': 'name1', 'id2': 'name2'}
        result = sort_user_id_name_map({'id2': 'name2', 'id1': 'name1'})
        self.assertEqual(result, {'id1': 'name1', 'id2': 'name2'})

if __name__ == '__main__':
    unittest.main()
```

En outre, lors d'une discussion entre l'utilisateur et Amazon Q Developer, celui-ci suggère et génère des tests unitaires basés sur les instructions de saisie du code de l'utilisateur. Pour plus d'informations, consultez la section Exemples de chat.

Java exemples

Procédez comme suit : Java des exemples montrent comment les utilisateurs peuvent commencer à générer des suggestions de code en ligne. Ces exemples illustratifs vous montrent comment

déclencher l'assistant de codage pour vos propres projets. La plupart des exemples suivants sont applicables à d'autres langues :

- Génération de classes et de fonctions
- Code du document
- Générer des algorithmes
- Générer des tests unitaires

Génération de classes et de fonctions

Pour générer une implémentation de classe complète ou partielle, utilisez des commentaires de code. Le commentaire décrit l'intention de la classe. Commencez simplement et ajoutez plus de détails si nécessaire. Après avoir généré une classe, vous pouvez continuer à générer des fonctions au sein de son squelette.

Lorsque vous rédigez des commentaires, il est préférable d'utiliser la structure de commentaires de la langue. Entrée Java, c'est `/** */`.

Entrée :

```
/**
 * DocumentReader class. has a name, children and optional parent.
 * Includes I/O functions to read and
 * write document
 */
```

Sortie :

```
/**
 * DocumentReader class. has a name, children and optional parent.
 * Includes I/O functions to read and
 * write document
 */
import java.util.ArrayList;
public class DocumentReader {
    private String name;
    private DocumentReader parent;
    private ArrayList<DocumentReader> children;

    /**
```

```
     * Constructor
     * @param name name of the document
     * @param parent parent of the document
     */
    public DocumentReader(String name, DocumentReader parent) {
        this.name = name;
        this.parent = parent;
        this.children = new ArrayList<>();
    }

    /**
     * Constructor
     * @param name name of the document
     */
    public DocumentReader(String name) {
        this(name, null);
    }
    /**
     * Getter for name
     * @return name of the document
     */
    public String getName() {
        return name;
    }

    /**
     * Getter for parent
     * @return parent of the document
     */
    public DocumentReader getParent() {
        return parent;
    }

    /**
     * Getter for children
     * @return children of the document
     */
    public ArrayList<DocumentReader> getChildren() {
        return children;
    }

    /**
     * Adds a child to the document
```

```
     * @param child child to be added
     */
    public void addChild(DocumentReader child) {
        children.add(child);
    }

    /**
     * Removes a child from the document
     * @param child child to be removed
     */
    public void removeChild(DocumentReader child) {
        children.remove(child);
    }
}
```

Maintenant qu'il existe une classe initiale, une fonction supplémentaire peut être générée avec des commentaires supplémentaires.

Entrée :

```
/**
* function to read document from local directory path
* and print first line of document
*/
```

Sortie :

```
/**
 * function to read document from local directory path
 * and print first line of document
 */
public void readDocument() {
    try {
        BufferedReader reader = new BufferedReader(new FileReader(name));
        String line = reader.readLine();
        System.out.println(line);
        reader.close();
    } catch (IOException e) {
        System.out.println("Error reading document: " + e.getMessage());
    }
}
```

Code du document

Réduisez le temps consacré à la documentation du code en vous appuyant sur la documentation en ligne. Ajoutez un Java pause dans les commentaires`/* */`.

Entrée :

```
import java.util.function.Function;

public class S3Utils {
    /*
     * <place cursor here>
    */
    public static Function<String, String> generateS3Function(String bucketName, String
 region) {
        return (String key) -> {
            String url = generateS3Url(bucketName, key, region);
            return url;
        };
    }
}
```

Si la génération en ligne n'a pas lieu, exécutez une suggestion en ligne d'Amazon Q Developer après avoir placé votre curseur à l'endroit où vous souhaitez obtenir de la documentation.

Sortie :

```
import java.util.function.Function;

public class S3Utils {
    /*
     * Generate a function that returns the S3 URL for a given key
     * @param bucketName the name of the bucket
     * @param region the region of the bucket
     * @return the function that generates the S3 URL
    */
    public static Function<String, String> generateS3Function(String bucketName, String
 region) {
        return (String key) -> {
            String url = generateS3Url(bucketName, key, region);
            return url;
        };
```

```
    }
}
```

Pour des explications plus simples du code (classes, fonctions ou éléments de script), utilisez des caractères directs (`//`) placés stratégiquement avant les sections de code dans lesquelles vous souhaitez ajouter de la documentation.

Entrée :

```
//
public class S3Utils {
    /*
     * Generate a function that returns the S3 URL for a given key
     * @param bucketName the name of the bucket
     * @param region the region of the bucket
     * @return the function that generates the S3 URL
    */
    public static Function<String, String> generateS3Function(String bucketName, String
 region) {
        return (String key) -> {
            String url = generateS3Url(bucketName, key, region);
            return url;
        };
    }
}
```

Sortie :

```
// This class is used to generate a function that returns the S3 URL for a given key
public class S3Utils {
    /*
     * Generate a function that returns the S3 URL for a given key
     * @param bucketName the name of the bucket
     * @param region the region of the bucket
     * @return the function that generates the S3 URL
    */
    public static Function<String, String> generateS3Function(String bucketName, String
 region) {
        return (String key) -> {
            String url = generateS3Url(bucketName, key, region);
            return url;
        };
```

```
    }
}
```

Générer des algorithmes

Amazon Q Developer peut générer des algorithmes populaires, notamment liés à AWS SDKs. Dans votre script, ajoutez une invite qui indique la fonction que vous souhaitez utiliser.

Cet exemple génère une carte de hachage des clés d'identification utilisateur vers les valeurs des noms, puis génère une fonction de tri à l'aide de l'algorithme de tri par bulles.

Entrée :

```
import java.util.HashMap;
public class SortList {

    // generate key value pair hash map for
    // user id and name
    public static HashMap<Integer, String> getUserIdNameMap() {
        HashMap<Integer, String> userIdNameMap = new HashMap<Integer, String>();
        userIdNameMap.put(1, "Amit");
        userIdNameMap.put(2, "Rahul");
        userIdNameMap.put(3, "Vijay");
        userIdNameMap.put(4, "Raj");
        userIdNameMap.put(5, "Kumar");
        return userIdNameMap;
    }

    /*
    * Create function to sort UserIdName map by name alphabetically using bubble sort
    */


}
```

Sortie :

```
/*
* Create function to sort UserIdName map by name alphabetically using bubble sort
*/
public static HashMap<Integer, String> sortUserIdNameMap(HashMap<Integer, String>
 userIdNameMap) {
```

```
    int size = userIdNameMap.size();
    for (int i = 0; i < size; i++) {
        for (int j = i + 1; j < size; j++) {
            if (userIdNameMap.get(i).compareTo(userIdNameMap.get(j)) > 0) {
                String temp = userIdNameMap.get(i);
                userIdNameMap.put(i, userIdNameMap.get(j));
                userIdNameMap.put(j, temp);
            }
        }
    }
    return userIdNameMap;
}
```

En outre, les utilisateurs peuvent créer des instructions pour les procédures standard. Dans cet exemple, l'utilisateur crée une invite pour stocker la carte de hachage dans une table Amazon DynamoDB. Au début du script, l'utilisateur ajoute une invite pour ajouter DynamoDB Java bibliothèques pour aider à fournir un contexte supplémentaire à Amazon Q Developer.

Entrée :

```
// add libraries for Amazon DynamoDB
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;

     /*
     * Store sorted UserIdName map into Amazon DynamoDB table named users
     */
```

Sortie :

```
// add libraries for Amazon DynamoDB
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;

     /*
     * Store sorted UserIdName map into Amazon DynamoDB table named users
     */
    public static void storeSortedUserIdNameMap(HashMap<Integer, String> userIdNameMap)
 {
        // create Amazon DynamoDB client
        AmazonDynamoDB client = AmazonDynamoDBClientBuilder.defaultClient();
```

```
        // store sorted UserIdName map into Amazon DynamoDB table named users
        for (int userId : userIdNameMap.keySet()) {
            String name = userIdNameMap.get(userId);
            client.putItem("users", new HashMap<String, Object>() {{
                put("userId", userId);
                put("name", name);
            }});
        }
    }
```

Générer des tests unitaires

Grâce aux commentaires intégrés, Amazon Q Developer peut générer des tests unitaires pour effectuer des assertions et d'autres JUnit tests. Dans cet exemple, l'utilisateur demande d'abord à Amazon Q de générer les bibliothèques applicables JUnit pour définir le contexte du script de test. Ensuite, l'utilisateur crée une classe publique suivie d'un déclarateur et de chaînes de commentaires.

Entrée :

```
// import libraries for java unit tests
import org.junit.Test;
import static org.junit.Assert.*;

public class UnitTests {

    @Test
    /*
     * Create unit test for getUserIdNameMap
     */
}
```

Sortie :

```
@Test
/*
* Create unit test for getUserIdNameMap
*/
public void testGetUserIdNameMap() {
    // create a new instance of the UserIdNameMap class
    UserIdNameMap userIdNameMap = new UserIdNameMap();
```

```
    // create a new instance of the User class
    User user = new User("John", "Doe");

    // add the user to the map
    userIdNameMap.addUser(user);

    // get the user from the map
    User retrievedUser = userIdNameMap.getUser("John");

    // assert that the retrieved user is the same as the user added to the map
    assertEquals(user, retrievedUser);
}
```

En outre, lors d'une discussion entre l'utilisateur et Amazon Q Developer, celui-ci suggère et génère des tests unitaires basés sur les instructions de saisie du code de l'utilisateur. Pour plus d'informations, consultez la section Exemples de chat.

Exemples de chat

Amazon Q Developer Chat aide les développeurs à effectuer diverses tâches, telles que le codage, le test et la mise à niveau des applications, le diagnostic des erreurs, la réalisation d'analyses de sécurité et de correctifs, ainsi que l'optimisation AWS des ressources. Cette section explique comment le chat avec Amazon Q peut aider un développeur à effectuer des tâches de programmation classiques, telles que les suivantes :

- Renseignez-vous sur Services AWS
- Générer du code
- Générer des tests unitaires
- Expliquer le code

Renseignez-vous sur Services AWS

Les utilisateurs peuvent poser des questions aux développeurs Amazon Q Services AWS, par exemple pour trouver le bon service ou comprendre les meilleures pratiques.

Discutez de Services AWS

Utilisateur :

Quel est le meilleur service de conteneur à utiliser pour gérer ma charge de travail si je dois réduire mes coûts ?

Amazon Q :

Sur la base des informations fournies, il semble que vous recherchiez un service de conteneur rentable pour gérer vos charges de travail. Dans ce cas, je recommande d'envisager Amazon Elastic Container Service (Amazon ECS) avec Fargate. [1]

Amazon ECS with Fargate est un moteur de calcul sans serveur pour les conteneurs qui vous permet d'exécuter vos conteneurs sans avoir à gérer d'infrastructure sous-jacente. Avec Fargate, vous n'avez pas besoin de provisionner ou de gérer des serveurs : Fargate s'en charge pour vous. Cela peut vous aider à réduire vos coûts, car vous ne payez que pour les ressources de calcul réellement utilisées par vos conteneurs, sans avoir à payer pour la capacité de serveur inactive. [2]

Voici quelques avantages clés liés à l'utilisation d'Amazon ECS avec Fargate :

• Pas besoin de gérer des serveurs ou des clusters : Fargate s'occupe du provisionnement et de la gestion de l'infrastructure pour vous.

• Pay-as-you-go tarification : vous ne payez que pour les ressources informatiques utilisées par vos conteneurs, sans avoir à payer pour les capacités inutilisées.

• Dimensionnement automatique — Fargate peut automatiquement redimensionner vos conteneurs à la hausse ou à la baisse en fonction de la demande.

• Intégration étroite avec d'autres outils Services AWS tels que l'équilibrage de charge, la journalisation et la surveillance.

Les ressources fournies mentionnent également que le service ECS offre une flexibilité en termes de planification et de placement des conteneurs, ce qui peut vous aider à optimiser davantage vos coûts.

Je vous recommande de consulter le détail des tarifs d'Amazon ECS with Fargate sur AWS le site Web afin d'avoir une meilleure idée des coûts en fonction de vos exigences spécifiques en matière de charge de travail. Et bien sûr, n'hésitez pas à me faire savoir si vous avez d'autres questions !

Sources

[1] Quel service de AWS conteneur vous convient le mieux ?

aptgetops.tech > - -vous aws-container-services-which one-is-right-for

[2] Java options de conteneur pour les charges de travail non HTTP

stackoverflow.com > Questions > 43417016

Mettre fin à la discussion sur Amazon Q Services AWS

Générer du code

Les utilisateurs peuvent demander de l'aide à Amazon Q Developer pour écrire des extraits de code.

Discuter de l'écriture de code

Utilisateur :

Pouvez-vous me donner un exemple de la façon de répertorier tous les compartiments S3 à l'aide du ? AWS SDK pour Python (Boto)

Amazon Q :

Voici certainement un exemple de la façon de répertorier tous les compartiments S3 à l'aide du AWS SDK pour Python (Boto)[1] :

```
import boto3
# Create an S3 client
s3 = boto3.client('s3')

# List all the buckets
response = s3.list_buckets()

# Print the name of each bucket
for bucket in response['Buckets']:
    print(bucket['Name'])
```

Voici comment fonctionne le code :

1. Nous importons d'abord la bibliothèque boto3, qui est le SDK pour Python.
2. Nous créons un client S3 en utilisant`boto3.client('s3')`.
3. Nous appelons la `list_buckets()` méthode sur le client S3, qui renvoie une réponse contenant une liste de tous les compartiments du compte.
4. Nous parcourons ensuite la `Buckets` liste dans la réponse et imprimons le nom de chaque compartiment.

Il s'agit d'un exemple simple, mais il AWS SDK pour Python (Boto) fournit de nombreuses autres fonctionnalités avancées pour travailler avec Amazon S3, telles que la création, la suppression et la gestion d'objets dans des compartiments, la définition de politiques de compartiment, etc. Vous trouverez d'autres exemples et de la documentation dans la documentation AWS SDK pour Python (Boto)(). [2]

N'oubliez pas que lorsque vous travaillez avec Services AWS, il est important de suivre les meilleures pratiques en matière de gestion des informations d'identification et des informations sensibles, telles que le stockage des clés AWS d'accès dans des variables d'environnement au lieu de les coder en dur dans le code.

Sources : [1] Exemples Amazon S3 utilisant le SDK pour Python (AWS Boto3) - Exemples de code SDK docs.aws.amazon.com > code-library > latest-1x >... > python-s3.html

[2] Commencez à utiliser les buckets et les objets Amazon S3 à l'aide d'un AWS SDK - Exemples de code AWS SDK docs.aws.amazon.com > code-library > latest-1x >... > s3_example_s3_buckets_ _sect... GettingStarted

Fin de la discussion Amazon Q sur l'écriture de code

Générer des tests unitaires

Les utilisateurs peuvent envoyer le code sélectionné sous forme d'invite à Amazon Q Developer, où des instructions supplémentaires sont appliquées, telles que la génération d'un test unitaire. Pour step-by-step obtenir des instructions, consultez la section Envoyer du code à Amazon Q dans le manuel Amazon Q Developer User Guide.

Discuter des tests unitaires

Utilisateur :

Rédigez un test unitaire avec des données fictives pour la fonction suivante :

```
// poll messages from Amazon SQS queue name
   public List<String> pollMessages(String queueName) {
       List<String> messages = new ArrayList<>();
       // poll messages from queue
       ReceiveMessageResult = sqsClient.receiveMessage(new ReceiveMessageRequest().
       for (Message : receiveMessageResult.getMessages()) {
           messages.add(message.getBody());
       }
```

```
        return messages;
        }
```

Amazon Q :

Pour écrire un test unitaire avec des données fictives pour la fonction PollMessages, nous pouvons utiliser un framework de simulation tel que Mockito. Voici un exemple :

```
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.model.Message;
import com.amazonaws.services.sqs.model.ReceiveMessageRequest;
import com.amazonaws.services.sqs.model.ReceiveMessageResult;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.mockito.Mockito;
import org.mockito.junit.jupiter.MockitoExtension;

import java.util.Arrays;
import java.util.List;

import static org.junit.jupiter.api.Assertions.assertEquals;

@ExtendWith(MockitoExtension.class)
public class SQSUtilsTest {

    @Mock
    private AmazonSQS sqsClient;

    @InjectMocks
    private SQSUtils sqsUtils;

    @Test
    public void testPollMessages() {
        // Arrange
        String queueName = "test-queue";
        Message message1 = new Message().withBody("message1");
        Message message2 = new Message().withBody("message2");
        ReceiveMessageResult receiveMessageResult = new
 ReceiveMessageResult().withMessages(message1, message2);
        Mockito.when(sqsClient.receiveMessage(new
 ReceiveMessageRequest().withQueueUrl(queueName)))
```

```
                .thenReturn(receiveMessageResult);

        // Act
        List<String> messages = sqsUtils.pollMessages(queueName);

        // Assert
        assertEquals(Arrays.asList("message1", "message2"), messages);
    }
}
```

Dans cet exemple, nous utilisons Mockito pour simuler le client AmazonSQS et configurer le comportement attendu de la méthode ReceiveMessage afin qu'elle renvoie deux messages fictifs dont les corps sont « message1 » et « message2 ». Ensuite, nous appelons la méthode PollMessages et affirmons que la liste de messages renvoyée correspond aux valeurs attendues.

Les points essentiels sont les suivants :

1. Nous utilisons l'annotation @ ExtendWith (MockitoExtension.class) pour activer l'intégration de Mockito avec. JUnit
2. Nous utilisons l'annotation @Mock pour créer une instance fictive du client AmazonSQS.
3. Nous utilisons l'InjectMocks annotation @ pour injecter le faux client AmazonSQS dans la SQSUtils classe.
4. Dans la méthode de test, nous avons défini le comportement attendu de la méthode ReceiveMessage à l'aide de Mockito.when () .thenReturn ().
5. Nous appelons la méthode PollMessages et affirmons que la liste des messages renvoyés correspond aux valeurs attendues.

Cette approche nous permet de tester la méthode PollMessages sans réellement interagir avec le service Amazon SQS, ce qui peut rendre le test plus rapide et plus fiable.

Fin de la discussion Amazon Q sur les tests unitaires

Expliquer le code

Pour les processus de révision du code, les développeurs peuvent envoyer une sélection de code à Amazon Q Developer afin de mieux comprendre le code. En outre, l'explication fournie par Amazon Q peut également servir à des fins de documentation. Dans l'exemple suivant, Amazon Q explique la fonction de gestion à l'aide d'une AWS Lambda fonction, puis propose une docstring.

Discuter sur l'explication du code

Utilisateur :

Expliquez la partie suivante de mon code :

```
def lambda_handler(event, context):
    dynamodb = boto3.resource('dynamodb')
    s3_client = boto3.client('s3')
    table = dynamodb.Table('DYNAMODB_TABLE')
    # Get s3 object from the event
    print(event)
    print(context)
    # get object from event
    bucket_key = event['Records'][0]['s3']['object']['key']
    object_key = unquote_plus(bucket_key)
    fullText = detect_text(object_key, bucket)

    response = s3_client.head_object(Bucket=bucket, Key=object_key)
    print(response)
    print(fullText)
    metadata = response['ResponseMetadata']
    metadata = json.dumps(metadata)

    now = datetime.now()
    current_time = now.strftime("%A %d %B, %Y %H:%M:%S")
    print("Current Time =", current_time + "\n\n")
    reason, outcome = detect_restaurant_closure(fullText, current_time)
    table.put_item(
        Item={
            'id': object_key,
            'creationTime': now.strftime("%Y-%m-%d %H:%M:%S"),
            'reason': reason,
            'text': fullText,
            'metadata': metadata
        }
    )
```

Amazon Q :

Le bloc de code fourni semble être une AWS Lambda fonction écrite en Python. Permettez-moi de vous expliquer les principaux aspects de ce code :

1. La fonction s'appelle lambda_handler et constitue le point d'entrée de la fonction Lambda.
2. Il utilise la bibliothèque boto3 pour interagir avec AWS DynamoDB et les services S3.
3. La fonction extrait une clé d'objet S3 et un nom de compartiment à partir des données d'événements entrantes.
4. Il appelle ensuite une fonction detect_text (qui n'apparaît pas dans le code fourni) pour extraire le contenu en texte intégral de l'objet S3.
5. La fonction récupère également les métadonnées relatives à l'objet S3 à l'aide de la méthode head_object.
6. Il extrait ensuite l'heure actuelle et appelle une fonction detect_restaurant_closure (également non illustrée) pour déterminer la raison et le résultat de la fermeture d'un restaurant sur la base du texte extrait.
7. Enfin, la fonction stocke les informations suivantes dans une table DynamoDB :
   - La clé de l'objet S3 comme identifiant
   - L'heure de création
   - La raison et le résultat de la fermeture du restaurant
   - Le texte extrait de l'objet S3

Dans l'ensemble, cette fonction Lambda semble faire partie d'un système plus vaste qui traite les objets S3, extrait le contenu du texte, l'analyse pour y détecter les informations relatives à la fermeture des restaurants et stocke les résultats dans une table DynamoDB. Les détails spécifiques des fonctions detect_text et detect_restaurant_closure seraient nécessaires pour bien comprendre les fonctionnalités de ce code.

Fin de la réponse d'Amazon Q

Utilisateur : résumez cette explication dans une docstring pour cette fonction lambda_handler

Amazon Q : Voici une chaîne de documentation suggérée pour la fonction lambda_handler [1] :

```
def lambda_handler(event, context):
    """
    AWS Lambda function that processes an S3 object, extracts text content, analyzes
 it, and stores the results.
    Args:
        event (dict): The event data that triggered the Lambda function, containing
 information about the S3 object.
```

```
        context (object): The context object provided by AWS Lambda, containing
information about the runtime environment.

    Returns:
        None
    """

    # function implementation details...
```

Cette docstring fournit un résumé de haut niveau de l'objectif de la fonction, des paramètres d'entrée attendus et de la valeur de retour attendue (ou de son absence, dans ce cas). Cela permet à l'utilisateur de comprendre clairement ce que fait la fonction lambda_handler sans entrer dans les détails de l'implémentation.

Sources : [1] Définissez le gestionnaire de fonctions Lambda en C# - AWS Lambda https://docs.aws.amazon.com/lambda/latest/dg/csharp-handler.html - : ~:text=Le gestionnaire de fonctions Lambda est, sort ou expire.

Fin de la discussion Amazon Q sur l'explication du code

Résolution des problèmes liés aux scénarios de génération de code dans Amazon Q Developer

Lorsque vous utilisez Amazon Q Developer, vous pouvez être confronté aux scénarios courants suivants, avec une génération de code et des résolutions inexactes :

- Génération de code vide
- Commentaires continus
- Génération de code en ligne incorrecte
- Résultats inadéquats des chats

Génération de code vide

Lors du développement du code, vous pouvez rencontrer les problèmes suivants :

- Amazon Q ne fournit aucune suggestion.
- Le message « Aucune suggestion de la part d'Amazon Q » apparaît dans votre IDE.

Vous pensez peut-être d'abord qu'Amazon Q ne fonctionne pas correctement. Cependant, la cause première de ces problèmes est généralement associée au contexte du script ou du projet ouvert dans l'IDE.

Si Amazon Q Developer ne fournit pas de suggestion automatiquement, vous pouvez utiliser les raccourcis suivants pour exécuter manuellement les suggestions en ligne d'Amazon Q :

- PC - Alt+C
- macOS - Option+C

Pour plus d'informations, consultez la section Utilisation des touches de raccourci dans le manuel Amazon Q Developer User Guide.

Dans la plupart des scénarios, Amazon Q génère une suggestion. Lorsqu'Amazon Q renvoie un message « Aucune suggestion de la part d'Amazon Q », passez en revue les solutions courantes suivantes pour ce problème :

- Contexte inadéquat — Assurez-vous que les bibliothèques applicables sont présentes. Assurez-vous que les classes et les fonctions définies dans le script sont liées au nouveau code.
- Demande ambiguë : si l'invite prête à confusion ou n'est pas claire, Amazon Q sera moins performant lorsqu'il proposera du code en ligne. Dans ces scénarios, soyez plus précis dans votre code et dans vos commentaires afin d'obtenir des réponses appropriées.

Si vous êtes bloqué, continuez à coder au mieux de vos capacités. En général, Amazon Q reconnaît votre intention, ce qui l'incite à proposer des suggestions. Une autre option consiste à envoyer un extrait de code avec des instructions rapides dans un chat Amazon Q. Pour plus d'informations, consultez la section Envoyer du code à Amazon Q dans le manuel Amazon Q Developer User Guide.

Commentaires continus

Vous pouvez rédiger une invite à l'aide de commentaires et ne recevoir que des lignes de commentaires génériques ou répétées de la part d'Amazon Q.

Entrée :

```
// Kafka test
```

Résultat :

```
// Kafka test
// Kafka test
// Kafka test
// Kafka test
```

Ce problème est généralement dû à un manque de contexte dans le script ou à de mauvaises instructions rapides. Nous vous recommandons d'utiliser le format standard pour les commentaires en fonction de la langue. Dans cet exemple, l'utilisateur a modifié l'invite à utiliser `/* */` pour un bloc de commentaires, et cette modification a amélioré les performances d'Amazon Q.

Entrée :

```
/*
 * Create a unit test to push a message to a Kafka topic
 */
```

Résultat :

Dans ce résultat, l'utilisateur a accepté plusieurs suggestions de manière séquentielle.

```
/*
 * Create a unit test to push a message to a Kafka topic
 */
import org.junit.Test;
import java.util.Properties;
import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.ProducerRecord;
import static org.junit.Assert.assertEquals;
// create function for a Kafka test
public class KafkaTest {
    @Test
    public void testKafka() {
        // create Kafka producer
        Properties props = new Properties();
        props.put("bootstrap.servers", "localhost:9092");
        props.put("key.serializer",
 "org.apache.kafka.common.serialization.StringSerializer");
        props.put("value.serializer",
 "org.apache.kafka.common.serialization.StringSerializer");
        KafkaProducer<String, String> producer = new KafkaProducer<>(props);
        // create Kafka topic
        String topic = "test";
        // send message to Kafka topic
        producer.send(new ProducerRecord<>(topic, "key", "value"));
        // close Kafka producer
        producer.close();
        // assert that message was sent successfully
        assertEquals("value", KafkaTest.consumer.poll().value());
    }
}
```

Génération de code en ligne incorrecte

Parfois, les suggestions en ligne d'Amazon Q Developer ne sont pas utiles ou ne répondent pas à l'objectif de votre invite. Parfois, les suggestions de code en ligne peuvent fournir une fonction générique similaire à une simple méthode Hello World.

Lorsque les suggestions de code intégrées ne sont pas utiles, essayez les meilleures pratiques suivantes :

- Passez en revue le contexte du script. En général, plus vous fournissez de contexte, meilleurs sont les résultats que vous obtenez avec Amazon Q.
- Si le contexte varie ou and/or est bruyant, modularisez votre code. Regroupez des fonctions similaires au sein d'un même script. Séparez les différentes classes et fonctions en fonction de leur objectif.
- Rédigez les noms de vos classes et de vos fonctions de manière tactique pour aider Amazon Q à fournir de meilleures recommandations.

Dans cet exemple, le développeur écrit un script à partir de zéro qui télécharge des fichiers sur Hadoop.

Entrée :

```
class  MyFunction{

    // create function to read data locally and upload to HDFS filesystem
```

Résultat :

```
class  MyFunction{

    // create function to read local data file path and upload to HDFS filesystem
    public void readAndUploadToHDFS(String inputPath, String outputPath) throws
 IOException {
        // read data from local file
        BufferedReader in = new BufferedReader(new FileReader(inputPath));

}
```

Bien qu'Amazon Q ait fourni une recommandation, celle-ci n'a pas répondu aux attentes du développeur. Pour améliorer les performances, le développeur importe des bibliothèques et modifie le nom de classe pour représenter les utilitaires Hadoop.

Correctif :

```
import java.io.BufferedReader;
```

```
import java.io.IOException;
import java.io.FileReader;

// import applicable hadoop libraries for IO operations
import org.apache.hadoop.fs.Path;

class  HadoopUtilities{

    // create function to read local data file path and upload to HDFS filesystem
}
```

Résultat :

Dans ce résultat, le développeur a accepté plusieurs suggestions de manière séquentielle.

```
class  HadoopUtilities{

    // create function to read local data file path and upload to HDFS filesystem
    public static void uploadToHDFS(String localFilePath, String hdfsFilePath) throws
 IOException {
        // create HDFS file path object
        Path hdfsPath = new Path(hdfsFilePath);
        // create local file path object
        Path localPath = new Path(localFilePath);
        // create hadoop file system object
        org.apache.hadoop.fs.FileSystem hdfs = org.apache.hadoop.fs.FileSystem.get(new
 org.apache.hadoop.conf.Configuration());
        // upload local file to HDFS
        hdfs.copyFromLocalFile(localPath, hdfsPath);
        // close HDFS file system
        hdfs.close();
        System.out.println("File uploaded successfully!");
}
```

Dans le même HadoopUtilities script, le développeur ajoute une invite de commentaire pour lire une rubrique relative à Amazon Simple Queue Service (Amazon SQS).

Entrée :

```
// read message from SQS
```

Résultat :

```
// read message from SQS
    public static void readMessageFromSQS(String message) {
        System.out.println("Received message from SQS: " + message);
```

Plutôt que d'utiliser des méthodes Java pour lire un message SQS depuis Amazon SQS, Amazon Q suggère d'imprimer une méthode sous forme de chaîne. Cependant, cette suggestion ne répond pas à l'objectif initial du développeur.

Dans ce cas, il est préférable de re-modulariser le code. Le script se concentre sur les méthodes Hadoop. Par conséquent, il peut être préférable de créer un script utilitaire spécialement conçu pour les opérations SQS.

La création d'une classe distincte pour les utilitaires SQS améliore la suggestion en ligne, qui peut être utilisée dans le HadoopUtilities script.

Entrée :

```
// import function for SQS
import com.amazonaws.services.sqs.AmazonSQS;
import java.util.ArrayList;
import java.util.List;


class SQSUtils {

    // instantiate SQS client
    private AmazonSQS sqsClient;

    // load SQS object
    public SQSUtils(AmazonSQS sqsClient) {
        this.sqsClient = sqsClient;
    }

    // poll message from Amazon SQS given queue name
}
```

Résultat :

```
// poll message from Amazon SQS given queue name
    public List<String> pollMessages(String queueName) {
```

```
        List<String> messages = new ArrayList<>();
        // poll messages from queue
        receiveMessageResult = sqsClient.receiveMessage(new
 ReceiveMessageRequest(queueName));
        for (Message message : receiveMessageResult.getMessages()) {
            messages.add(message.getBody());
        }
        return messages;
```

Si la suggestion en ligne ne donne toujours pas les résultats escomptés, demandez à Amazon Q Developer de générer du code lors d'une session de chat.

Discuter de la génération de code

Utilisateur :

Génération d'une fonction Java pour lire un message depuis une file d'attente Amazon SQS

Amazon Q :

Voici un exemple de fonction Java permettant de lire un message depuis une file d'attente Amazon SQS :

```
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.model.Message;
import com.amazonaws.services.sqs.model.ReceiveMessageRequest;
import com.amazonaws.services.sqs.model.ReceiveMessageResult;
public class SQSMessageReader {
    private AmazonSQS sqsClient;
    private String queueUrl;

    public SQSMessageReader(AmazonSQS sqsClient, String queueUrl) {
        this.sqsClient = sqsClient;
        this.queueUrl = queueUrl;
    }

    public String readMessage() {
        ReceiveMessageRequest receiveMessageRequest = new
 ReceiveMessageRequest(queueUrl)
                .withMaxNumberOfMessages(1)
                .withWaitTimeSeconds(10);
```

```
        ReceiveMessageResult receiveMessageResult =
sqsClient.receiveMessage(receiveMessageRequest);

        if (receiveMessageResult.getMessages().size() > 0) {
            Message message = receiveMessageResult.getMessages().get(0);
            String messageBody = message.getBody();
            String receiptHandle = message.getReceiptHandle();

            // Process the message
            System.out.println("Received message: " + messageBody);

            // Delete the message from the queue
            sqsClient.deleteMessage(queueUrl, receiptHandle);

            return messageBody;
        } else {
            return null;
        }
    }
```

Ce code crée une classe SQSMessage Reader qui prend un client AmazonSQS et une URL de file d'attente comme paramètres de constructeur. La méthode readMessage () utilise le ReceiveMessageRequest pour extraire jusqu'à 1 message de la file d'attente, avec un temps d'attente maximal de 10 secondes.

Si un message est reçu, le corps du message est imprimé et le message est supprimé de la file d'attente à l'aide du DeleteMessageRequest. Le corps du message est ensuite renvoyé. Si aucun message n'est reçu, nul est renvoyé.

Mettre fin à la discussion Amazon Q sur la génération de code

Résultats inadéquats des chats

Lors du développement du code, les développeurs peuvent demander à Amazon Q de générer des fonctions communes, de fournir des recommandations ou d'expliquer le code. Parfois, à la suite d'une demande (telle qu'une question ou un extrait de code), Amazon Q peut fournir des commentaires généraux ou des résultats qui ne répondent pas à vos attentes. Dans ces scénarios, essayez ce qui suit :

- Testez différentes instructions pour améliorer les résultats de génération de code fournis par Amazon Q.

- Suivez URLs les instructions fréquemment fournies par Amazon Q dans ses réponses à vos questions ou à vos questions.
- Faites-nous part de vos commentaires afin d'améliorer les performances futures d'Amazon Q.

Par exemple, un développeur est en train de créer une solution RAG (Retrieval Augmented Generation) et il doit fragmenter son document en fonction des relations sémantiques entre les phrases. Le développeur discute avec Amazon Q comme suit.

Discuter d'une solution RAG

Utilisateur :

Pour une solution RAG, générez du code pour le découpage sémantique

Amazon Q :

Voici certainement un exemple de la façon dont vous pourriez implémenter le découpage sémantique pour une solution de génération augmentée de récupération (RAG) en Python utilisant la bibliothèque : LangChain

```
from langchain.text_splitter import CharacterTextSplitter, TokenTextSplitter
from langchain.embeddings import HuggingFaceEmbeddings
from langchain.vectorstores import Chroma

# Initialize the text splitter
text_splitter = TokenTextSplitter(
    chunk_size=500,
    chunk_overlap=50,
    length_function=len
)
# Initialize the embeddings model
embeddings = HuggingFaceEmbeddings()

# Create the vector store
vectorstore = Chroma(
    embedding_function=embeddings.embed_documents,
    persist_directory="./chroma-data"
)

# Chunk the text using semantic similarity
def semantic_chunk(text):
```

```
    chunks = text_splitter.split_text(text)
    embeddings = embeddings.embed_documents(chunks)
    vectorstore.add_texts(chunks, embeddings)
    return chunks
```

Dans cet exemple, nous utilisons le formulaire TokenTextSplitter LangChain pour diviser le texte saisi en blocs de 500 caractères avec un chevauchement de 50 caractères. Le HuggingFaceEmbeddings modèle est utilisé pour générer des intégrations pour chaque segment, et le magasin Chroma vectoriel est utilisé pour stocker les fragments et leurs intégrations.

La fonction semantic_chunk prend le texte saisi, le divise en morceaux à l'aide du text_splitter, génère des intégrations pour chaque segment à l'aide du modèle d'intégration, puis ajoute les morceaux et leurs intégrations au. vectorstore

Cette approche vous permet de diviser le texte d'entrée en segments sémantiquement significatifs, ce qui peut être utile pour diverses tâches, telles que la réponse à des questions ou la synthèse, dans une solution RAG.

Notez qu'il s'agit d'un exemple simple et que vous devrez peut-être ajuster les paramètres (par exemple, chunk_size, chunk_overlap) ou le choix du séparateur de texte et du modèle d'intégration en fonction de votre cas d'utilisation et de vos données spécifiques.

Fin de la discussion Amazon Q à propos d'une solution RAG

Toutefois, cette approche ne fournit qu'un moyen simple de fragmenter des documents en utilisant la technique de la fenêtre coulissante. Le développeur souhaite une méthode avancée qui repose sur la mesure des intégrations. Dans ces scénarios, le développeur doit ajouter plus de détails à l'invite en fonction de son objectif. Lorsque le développeur fait référence à la fonction connue de séparation de AI21 texte avec LangChain, il reçoit une réponse améliorée, comme indiqué dans le chat suivant.

Discuter d'une solution RAG (avec une invite améliorée)

Utilisateur :

L'invite ci-dessus ne concerne que le fractionnement en fonction de la taille du jeton par morceau. Je souhaite utiliser un séparateur de texte AI21 sémantique. Générez une fonction en utilisant le découpage sémantique de texte A121 avec langchain.

Amazon Q :

OK, utilisons le séparateur de texte AI21 sémantique LangChain pour implémenter le découpage sémantique. Voici un exemple : [1]

```
from langchain.text_splitter import AI21SemanticTextSplitter
from langchain.embeddings import HuggingFaceEmbeddings
from langchain.vectorstores import Chroma

# Initialize the text splitter
text_splitter = AI21SemanticTextSplitter()

# Initialize the embeddings model
embeddings = HuggingFaceEmbeddings()

# Create the vector store
vectorstore = Chroma(
    embedding_function=embeddings.embed_documents,
    persist_directory="./chroma-data"
)

# Chunk the text using semantic similarity
def semantic_chunk(text):
    chunks = text_splitter.split_text(text)
    embeddings = embeddings.embed_documents(chunks)
    vectorstore.add_texts(chunks, embeddings)
    return chunks
```

Dans cet exemple, nous utilisons le formulaire AI21 SemanticTextSplitter LangChain pour diviser le texte saisi en segments sémantiquement significatifs. Le HuggingFaceEmbeddings modèle est utilisé pour créer des intégrations pour chaque segment, et le magasin Chroma vectoriel est utilisé pour stocker les fragments et leurs intégrations.

La fonction semantic_chunk prend le texte saisi, le divise en morceaux à l'aide du text_splitter, crée des intégrations pour chaque segment à l'aide du modèle d'intégration, puis ajoute les morceaux et leurs intégrations au. vectorstore

Cette approche vous permet de diviser le texte d'entrée en segments sémantiquement significatifs, ce qui peut être utile pour diverses tâches, telles que la réponse à des questions ou la synthèse, dans une solution RAG.

Veuillez noter que vous devrez peut-être ajuster les paramètres ou le choix du modèle d'intégration en fonction de votre cas d'utilisation et de vos données spécifiques.

Sources [1] Apprenez à utiliser les séparateurs de texte dans LangChain

soosweb3.hashnode.dev > -séparateurs de texte the-ultimate-langchain-series

Fin de la discussion Amazon Q à propos d'une solution RAG (avec invite améliorée)

Sur la base des résultats du chat, le développeur peut utiliser la fonction et accéder à la source des informations.

En général, les sujets et questions avancés nécessitent un contexte plus spécifique lorsque vous discutez avec Amazon Q Developer. Si vous pensez que les résultats de votre chat ne sont pas exacts, utilisez l'icône représentant un pouce vers le bas pour nous faire part de vos commentaires sur la réponse d'Amazon Q. Amazon Q Developer utilise en permanence les commentaires pour améliorer les futures versions. Pour les interactions qui ont produit des résultats positifs, il est utile de fournir vos commentaires à l'aide de l'icône représentant le pouce levé.

FAQs à propos d'Amazon Q Developer

Cette section fournit des réponses aux questions fréquemment posées concernant l'utilisation d'Amazon Q Developer pour le développement de code.

Qu’est-ce qu’Amazon Q Developer ?

Amazon Q Developer est un puissant service génératif basé sur l'IA conçu pour accélérer les tâches de développement de code en fournissant une génération de code intelligente et des recommandations. Le 30 avril 2024, Amazon a CodeWhisperer rejoint Amazon Q Developer.

Comment accéder à Amazon Q Developer ?

Amazon Q Developer est disponible dans le cadre des AWS boîtes à outils pour Visual Studio Code JetBrains IDEs, telles que PyCharm IntelliJ et. Pour commencer, installez la dernière AWS Toolkit version.

Quels sont les langages de programmation pris en charge par Amazon Q Developer ?

Pour Visual Studio Code et JetBrains IDEs Amazon Q Developer prend en charge PythonJava, JavaScript, TypeScript,, C#, GoRust, PHP, Ruby Kotlin, C, C++, les scripts Shell, SQL et Scala. Bien que ce guide se concentre sur Python et à titre d'Javaexemple, les concepts sont applicables à tous les langages de programmation pris en charge.

Comment puis-je fournir du contexte à Amazon Q Developer pour une meilleure génération de code ?

Commencez par le code existant, importez les bibliothèques pertinentes, créez des classes et des fonctions ou établissez des squelettes de code. Utilisez des blocs de commentaires standard pour les instructions en langage naturel. Concentrez votre script sur des objectifs spécifiques et modularisez les fonctionnalités distinctes dans des scripts distincts avec un contexte pertinent. Pour plus d'informations, consultez la section Meilleures pratiques de codage avec Amazon Q Developer.

Que dois-je faire si la génération de code en ligne avec Amazon Q Developer n'est pas précise ?

Vérifiez le contexte du script, assurez-vous que les bibliothèques sont présentes et que les classes et les fonctions sont liées au nouveau code. Modularisez votre code et séparez les différentes classes et fonctions en fonction de leur objectif. Rédigez des instructions ou des commentaires clairs et précis. Si vous n'êtes toujours pas sûr de l'exactitude du code et que vous ne pouvez pas continuer, lancez une discussion avec Amazon Q et envoyez-lui l'extrait de code avec les instructions. Pour plus d'informations, consultez la section Résolution des problèmes liés aux scénarios de génération de code dans Amazon Q Developer.

Comment puis-je utiliser la fonctionnalité de chat Amazon Q Developer pour générer du code et résoudre les problèmes ?

Discutez avec Amazon Q pour générer des fonctions communes, demander des recommandations ou expliquer le code. Si la réponse initiale n'est pas satisfaisante, testez différentes instructions et suivez les instructions fournies URLs. Envoyez également des commentaires à Amazon Q afin de l'aider à améliorer ses futures performances de chat. Utilisez les icônes du pouce vers le haut et du pouce vers le bas pour nous faire part de vos commentaires. Pour plus d'informations, consultez la section Exemples de chat.

Quelles sont les meilleures pratiques pour utiliser Amazon Q Developer ?

Fournissez un contexte pertinent, testez et répétez les instructions, examinez les suggestions de code avant de les accepter, utilisez les fonctionnalités de personnalisation et comprenez les politiques de confidentialité des données et d'utilisation du contenu. Pour plus d'informations, consultez les meilleures pratiques pour la génération de code avec Amazon Q Developer et les meilleures pratiques pour les recommandations de code avec Amazon Q Developer.

Puis-je personnaliser Amazon Q Developer pour générer des recommandations basées sur mon propre code ?

Oui, utilisez les personnalisations, qui constituent une fonctionnalité avancée d'Amazon Q Developer. Grâce aux personnalisations, les entreprises peuvent fournir leurs propres référentiels de code pour

permettre à Amazon Q Developer de recommander des suggestions de code en ligne. Pour plus d'informations, consultez la section Fonctionnalités avancées d'Amazon Q Developer and Resources.

Prochaines étapes de l'utilisation d'Amazon Q Developer

Grâce aux connaissances acquises grâce à ce guide complet, vous pouvez utiliser Amazon Q Developer de manière efficace dans votre flux de travail de codage. Installez-le AWS Toolkit dans votre IDE préféré (Visual Studio Code ou JetBrains) et commencez à explorer la génération de code générative basée sur l'IA et les recommandations d'Amazon Q Developer.

Le moyen le plus efficace d'exploiter tout le potentiel d'Amazon Q Developer consiste à acquérir une expérience pratique avec votre propre code. Lorsque vous intégrez Amazon Q à votre cycle de développement, consultez ce guide pour découvrir les meilleures pratiques, le dépannage et des exemples concrets.

En outre, tenez-vous au courant en consultant les AWS blogs et les guides de développement référencés dans Ressources. Ces ressources fournissent les dernières mises à jour, les meilleures pratiques et les informations nécessaires pour vous aider à optimiser votre utilisation d'Amazon Q Developer.

Vos commentaires sont précieux pour améliorer ce guide et l'aider à rester une ressource précieuse pour les développeurs. Partagez vos expériences, vos défis et vos suggestions pour les versions futures. Vos commentaires permettront d'améliorer le guide avec des exemples supplémentaires, des scénarios de résolution des problèmes et des informations adaptées à vos besoins.

Ressources

AWS blogues

- Accélération du cycle de développement de vos logiciels avec Amazon Q
- Réinventer le développement logiciel avec l'agent de développement Amazon Q
- Cinq exemples de résolution des problèmes avec Amazon Q
- Personnalisez Amazon Q Developer dans votre environnement IDE grâce à votre base de code privée
- Trois manières dont l'agent Amazon Q Developer pour la transformation du code accélère les mises à niveau de Java
- Tirer parti d'Amazon Q Developer pour un débogage et une maintenance de code efficaces
- Tester vos applications avec Amazon Q Developer

AWS documentation

- Guide de l'utilisateur Amazon Q pour les développeurs
- Personnalisation du code Amazon Q Developer
- Transformation du code Amazon Q Developer

AWS ateliers

- Journée d'immersion pour développeurs Amazon Q
- Atelier pour développeurs Amazon Q - Création de l'application Q-Words
- Atelier pour développeurs Amazon Q - Création d'instructions efficaces

Collaborateurs

Les personnes suivantes ont contribué à ce guide :

- Joe King, data scientist senior, AWS
- Prateek Gupta, chef d'équipe — Sr. CAA, AWS
- Manohar Reddy Arranagu, architecte, DevOps AWS
- Soumik Roy, architecte d'applications cloud, AWS
- Sanket Shinde, consultante, AWS

Historique du document

Le tableau suivant décrit les modifications importantes apportées à ce guide. Pour être averti des mises à jour à venir, abonnez-vous à un RSSfil RSS.

Modification	Description	Date
Publication initiale	—	16 août 2024

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.