



Praktik terbaik dengan Amazon Q Developer untuk pembuatan kode in-line dan asisten

# AWS Bimbingan Preskriptif



# AWS Bimbingan Preskriptif: Praktik terbaik dengan Amazon Q Developer untuk pembuatan kode in-line dan asisten

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

Merek dagang dan tampilan dagang Amazon tidak boleh digunakan sehubungan dengan produk atau layanan apa pun yang bukan milik Amazon, dengan cara apa pun yang dapat menyebabkan kebingungan di antara pelanggan, atau dengan cara apa pun yang merendahkan atau mendiskreditkan Amazon. Semua merek dagang lain yang tidak dimiliki oleh Amazon merupakan hak milik masing-masing pemiliknya, yang mungkin atau mungkin tidak terafiliasi, terkait dengan, atau disponsori oleh Amazon.

---

# Table of Contents

Pengantar .....	1
Tujuan .....	1
Alur kerja Developer .....	3
Desain dan perencanaan .....	4
Pengkodean .....	4
Tinjauan kode .....	5
Integrasi dan Deployment .....	5
Kemampuan tingkat lanjut .....	6
Transformasi kode Pengembang Amazon Q .....	6
Kustomisasi Pengembang Amazon Q .....	6
Praktik pengkodean terbaik .....	8
Praktik terbaik untuk orientasi .....	8
Prasyarat untuk Pengembang Amazon Q .....	8
Praktik terbaik saat menggunakan Amazon Q Developer .....	8
Privasi data dan penggunaan konten di Amazon Q Developer .....	9
Praktik terbaik untuk pembuatan kode .....	9
Praktik terbaik untuk rekomendasi kode .....	10
Contoh kode .....	12
Python contoh .....	12
Menghasilkan kelas dan fungsi .....	12
Kode dokumen .....	14
Menghasilkan algoritma .....	15
Hasilkan pengujian unit .....	17
Java contoh .....	18
Menghasilkan kelas dan fungsi .....	19
Kode dokumen .....	21
Menghasilkan algoritma .....	23
Hasilkan pengujian unit .....	26
Contoh obrolan .....	27
Tanyakan tentang Layanan AWS .....	27
Hasilkan kode .....	29
Hasilkan pengujian unit .....	30
Jelaskan kode .....	32
Pemecahan masalah .....	36

Pembuatan kode kosong .....	36
Komentar terus menerus .....	37
Pembuatan kode in-line salah .....	38
Hasil yang tidak memadai dari obrolan .....	43
FAQs .....	48
Apa itu Pengembang Amazon Q? .....	48
Bagaimana cara mengakses Amazon Q Developer? .....	48
Bahasa pemrograman apa yang didukung Pengembang Amazon Q? .....	48
Bagaimana saya bisa memberikan konteks kepada Pengembang Amazon Q untuk pembuatan kode yang lebih baik? .....	48
Apa yang harus saya lakukan jika pembuatan kode in-line dengan Pengembang Amazon Q tidak akurat? .....	49
Bagaimana cara menggunakan kemampuan obrolan Amazon Q Developer untuk pembuatan kode dan pemecahan masalah? .....	49
Apa saja praktik terbaik untuk menggunakan Amazon Q Developer? .....	49
Dapatkah saya menyesuaikan Pengembang Amazon Q untuk menghasilkan rekomendasi berdasarkan kode saya sendiri? .....	49
Langkah selanjutnya .....	51
Sumber daya .....	52
AWS blog .....	52
AWS dokumentasi .....	52
AWS lokakarya .....	52
Kontributor .....	53
Riwayat dokumen .....	54
Glosarium .....	55
# .....	55
A .....	56
B .....	59
C .....	61
D .....	64
E .....	68
F .....	70
G .....	72
H .....	73
I .....	74
L .....	77

---

M .....	78
O .....	83
P .....	85
Q .....	88
R .....	89
D .....	92
T .....	96
U .....	97
V .....	98
W .....	98
Z .....	99
.....	ci

# Praktik terbaik dengan Amazon Q Developer untuk pembuatan kode in-line dan asisten

Amazon Web Services ([Kontributor](#))

Agustus 2024 () [Riwayat dokumen](#)

Secara tradisional, pengembang mengandalkan keahlian, dokumentasi, dan cuplikan kode mereka sendiri dari berbagai sumber untuk menulis dan memelihara kode. Meskipun metode ini telah melayani industri dengan baik, mereka dapat memakan waktu dan rentan terhadap kesalahan manusia, yang menyebabkan inefisiensi dan potensi bug.

Di sinilah Pengembang Amazon Q masuk untuk meningkatkan perjalanan pengembang. Amazon Q Developer adalah asisten bertenaga AI AWS generatif yang kuat yang dirancang untuk mempercepat tugas pengembangan kode dengan menyediakan pembuatan dan rekomendasi kode cerdas.

Namun, seperti halnya teknologi baru lainnya, bisa ada tantangan. Harapan yang tidak realistis, kesulitan orientasi, pemecahan masalah pembuatan kode yang tidak akurat, dan menggunakan kemampuan Amazon Q dengan benar adalah rintangan umum yang mungkin dihadapi pengembang. Panduan komprehensif ini membahas tantangan ini, menyediakan skenario kehidupan nyata, praktik terbaik terperinci, pemecahan masalah, dan contoh kode kehidupan nyata praktis khusus untuk Python dan Java, dua bahasa pemrograman yang paling banyak diadopsi.

Panduan ini berfokus pada penggunaan Amazon Q Developer untuk melakukan tugas pengembangan kode seperti:

- Penyelesaian kode — Hasilkan saran in-line sebagai kode pengembang secara real time.
- Perbaiki dan saran kode — Diskusikan pengembangan perangkat lunak, buat kode baru dengan bahasa alami, dan tingkatkan kode yang ada.

## Tujuan

Tujuan panduan ini adalah untuk mendukung pengembang yang merupakan pengguna baru atau berkelanjutan dari Pengembang Amazon Q, membantu mereka untuk menggunakan layanan dengan sukses dalam tugas pengkodean sehari-hari mereka. Manajer tim pengembangan juga dapat mengambil manfaat dari membaca panduan ini.

---

Panduan ini memberi Anda wawasan berikut tentang penggunaan Amazon Q Developer:

- Memahami penggunaan Amazon Q Developer yang efektif untuk pengembangan kode
  - Berikan praktik terbaik untuk mengintegrasikan Amazon Q Developer ke dalam [alur kerja pengembang](#).
  - Tawarkan step-by-step panduan dengan contoh untuk [pembuatan kode](#) dan [rekomendasi](#) yang sukses.
- Mengurangi tantangan umum dan mempromosikan kejelasan pengembang untuk menggunakan Amazon Q Developer
  - Menawarkan [strategi](#) dan wawasan untuk memenuhi harapan pengembang dan mengatasi rintangan yang terkait dengan akurasi dan kinerja pembuatan kode.
- Menyediakan pemecahan masalah dan penanganan kesalahan
  - Lengkapi pengembang dengan [panduan pemecahan masalah](#) pembuatan kode Pengembang Amazon Q untuk mengatasi hasil yang tidak akurat atau perilaku yang tidak terduga.
  - Berikan [contoh dan skenario](#) kehidupan nyata yang spesifik untuk Python dan Java.
- Optimalkan alur kerja dan produktivitas
  - Optimalkan alur kerja pengembangan kode dengan Amazon Q Developer.
  - Diskusikan strategi untuk meningkatkan [produktivitas pengembang](#).

# Menggunakan Amazon Q Developer dalam alur kerja pengembang

Pengembang mengikuti alur kerja standar yang mencakup tahapan pengumpulan persyaratan, desain dan perencanaan, pengkodean, pengujian, tinjauan kode, dan penerapan. Bagian ini berfokus pada bagaimana Anda dapat menggunakan kemampuan Pengembang Amazon Q untuk mengoptimalkan langkah-langkah pengembangan utama.

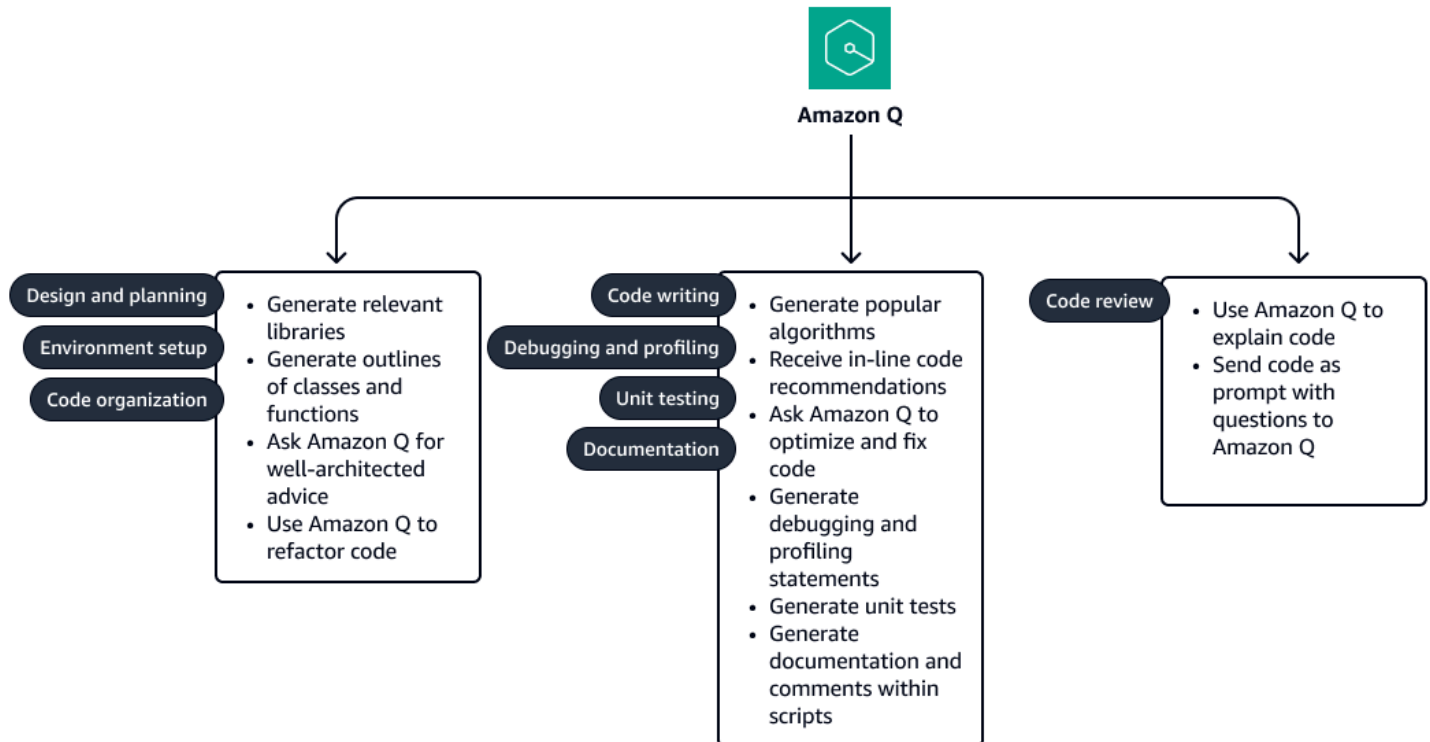


Diagram sebelumnya menunjukkan bagaimana Amazon Q Developer dapat mempercepat dan merampingkan tugas-tugas umum berikut dalam tahap pengembangan kode:

- Desain dan perencanaan | Pengaturan lingkungan | Kode organisasi
  - Hasilkan perpustakaan yang relevan
  - Hasilkan garis besar kelas dan fungsi
  - Mintalah Amazon Q untuk saran yang dirancang dengan baik
  - Gunakan Amazon Q untuk memfaktorkan ulang kode
- Penulisan kode | Debugging dan profiling | Pengujian unit | Dokumentasi
  - Hasilkan algoritme populer

- Terima rekomendasi kode in-line
- Minta Amazon Q untuk mengoptimalkan dan memperbaiki kode
- Hasilkan pernyataan debugging dan profiling
- Hasilkan tes unit
- Hasilkan dokumentasi dan komentar dalam skrip
- Tinjauan kode
  - Minta Amazon Q untuk menjelaskan kode
  - Kirim kode secepatnya dengan pertanyaan ke Amazon Q

## Desain dan perencanaan

Setelah mengumpulkan persyaratan bisnis dan teknis, pengembang merancang basis kode baru, atau memperluas basis kode yang sudah ada. Selama fase ini, Amazon Q Developer dapat membantu pengembang untuk melakukan tugas-tugas berikut:

- Hasilkan perpustakaan yang relevan dan garis besar kelas dan fungsi untuk saran yang dirancang dengan baik.
- Memberikan panduan untuk pertanyaan teknik, kompatibilitas, dan desain arsitektur.

## Pengkodean

Proses pengkodean menggunakan Amazon Q Developer untuk mempercepat pengembangan dengan cara berikut:

- Pengaturan lingkungan - Instal AWS Toolkit di lingkungan pengembangan terintegrasi Anda (IDE) (misalnya, VS Code atau IntelliJ). Kemudian, gunakan Amazon Q untuk membuat pustaka atau menerima saran penyiapan berdasarkan sasaran proyek Anda. Untuk detail selengkapnya, lihat [Praktik terbaik untuk orientasi Pengembang Amazon Q](#).
- Organisasi kode - Refactor kode atau dapatkan rekomendasi organisasi dari Amazon Q yang selaras dengan tujuan proyek Anda.
- Penulisan kode - Gunakan saran in-line untuk menghasilkan kode saat mengembangkan atau meminta Amazon Q untuk menghasilkan kode dengan menggunakan panel obrolan Amazon Q di AndaiDE. Untuk detail selengkapnya, lihat [Praktik terbaik untuk pembuatan kode dengan Amazon Q Developer](#).

- Debugging dan profiling - Buat perintah pembuatan profil, atau gunakan opsi Amazon Q seperti Perbaiki dan Jelaskan untuk men-debug masalah.
- Pengujian unit - Berikan kode sebagai prompt ke Amazon Q selama sesi obrolan dan minta pembuatan pengujian unit yang berlaku. Untuk informasi selengkapnya, lihat [Contoh kode dengan Developer Amazon Q](#).
- Dokumentasi - Gunakan saran in-line untuk membuat komentar dan docstring, atau gunakan opsi Jelaskan untuk menghasilkan ringkasan terperinci untuk pemilihan kode. Untuk informasi selengkapnya, lihat [Contoh kode dengan Developer Amazon Q](#).

## Tinjauan kode

Reviewer perlu memahami kode pengembangan sebelum mempromosikannya ke produksi. Untuk mempercepat proses ini, gunakan opsi Amazon Q Jelaskan dan Optimalkan, atau kirim pilihan kode dengan instruksi prompt khusus ke Amazon Q dalam sesi obrolan. Untuk informasi selengkapnya, lihat [Contoh obrolan](#).

## Integrasi dan Deployment

Mintalah Amazon Q untuk panduan tentang integrasi berkelanjutan, jalur pengiriman, dan praktik terbaik penerapan yang khusus untuk arsitektur proyek Anda.

Dengan menggunakan rekomendasi ini, Anda dapat belajar memanfaatkan fitur Pengembang Amazon Q secara efektif, mengoptimalkan alur kerja, dan meningkatkan produktivitas di seluruh siklus pengembangan.

# Kemampuan lanjutan dari Amazon Q Developer

Meskipun panduan ini berfokus pada penggunaan Amazon Q Developer dalam tugas pemrograman langsung, penting untuk mengetahui kemampuannya berikut:

- Transformasi kode Pengembang Amazon Q
- Kustomisasi Pengembang Amazon Q

## Transformasi kode Pengembang Amazon Q

Agen Pengembang Amazon Q untuk transformasi kode dapat meningkatkan versi bahasa kode file Anda tanpa perlu menulis ulang kode secara manual. Ini bekerja dengan menganalisis file kode yang ada dan secara otomatis menulis ulang untuk menggunakan versi bahasa yang lebih baru. Misalnya, Amazon Q mengubah satu modul jika Anda bekerja di IDE seperti Eclipse. Jika Anda menggunakan Visual Studio Code, Amazon Q dapat mengubah seluruh proyek atau ruang kerja.

Gunakan Amazon Q saat Anda ingin melakukan tugas pemutakhiran kode umum seperti berikut ini:

- Perbarui kode untuk bekerja dengan sintaks baru dari versi bahasa.
- Jalankan pengujian unit untuk memvalidasi kompilasi dan eksekusi yang berhasil.
- Periksa dan selesaikan masalah penerapan.

Amazon Q dapat menyelamatkan pengembang dari hari ke bulan pekerjaan yang membosankan dan berulang untuk meningkatkan basis kode.

Per Juni 2024, Pengembang Amazon Q mendukung peningkatan Java kode dan dapat mengubah Java 8 kode ke versi yang lebih baru seperti Java 11 atau 17.

## Kustomisasi Pengembang Amazon Q

Dengan kemampuan kustomisasi, Amazon Q Developer dapat memberikan saran in-line berdasarkan basis kode perusahaan sendiri. Perusahaan menyediakan repositori kode mereka baik ke Amazon Simple Storage Service (Amazon S3) atau AWS CodeConnections melalui, sebelumnya dikenal sebagai Connections. AWS CodeStar Kemudian, Amazon Q menggunakan repositori kode khusus dengan keamanan yang diaktifkan untuk merekomendasikan pola pengkodean yang relevan dengan pengembang di organisasi itu.

---

Saat menggunakan kustomisasi Amazon Q Developer, perhatikan hal-hal berikut:

- Pada Juni 2024, fitur Kustomisasi Pengembang Amazon Q dalam mode pratinjau. Akibatnya, fitur tersebut mungkin terbatas dalam ketersediaan dan dukungan.
- Saran kode in-line khusus hanya akan akurat mengingat kualitas repositori kode yang disediakan. Kami menyarankan Anda meninjau [skor evaluasi](#) untuk setiap penyesuaian yang Anda buat.
- Untuk mengoptimalkan kinerja, kami sarankan Anda menyertakan setidaknya 20 file data yang berisi bahasa yang diberikan di mana semua file sumber lebih besar dari 10MB. Pastikan repositori Anda terdiri dari kode sumber yang dapat direferensikan dan bukan file metadata (misalnya, file konfigurasi, file properti, dan file readme.)

Dengan menggunakan kustomisasi Amazon Q Developer, Anda dapat menghemat waktu dengan cara berikut:

- Gunakan rekomendasi yang didasarkan pada kode kepemilikan perusahaan Anda sendiri.
- Meningkatkan kegunaan kembali basis kode yang ada.
- Buat pola berulang yang digeneralisasikan di seluruh perusahaan Anda.

# Praktik pengkodean terbaik dengan Amazon Q Developer

Bagian ini membahas praktik terbaik untuk pengkodean dengan Amazon Q Developer. Praktik terbaik meliputi kategori berikut:

- [Orientasi](#) - Metode dan pertimbangan saat orientasi
- [Pembuatan kode](#) - Panduan untuk menggunakan pembuatan kode dengan sukses
- [Rekomendasi kode](#) - Teknik untuk meningkatkan kode

## Praktik terbaik untuk onboarding Amazon Q Developer

Amazon Q Developer adalah asisten pengkodean AI generatif yang kuat yang tersedia melalui populer IDEs seperti Visual Studio Code dan JetBrains. Bagian ini berfokus pada praktik terbaik untuk mengakses dan mengarahkan Pengembang Amazon Q ke lingkungan pengembangan pengkodean Anda.

## Prasyarat untuk Pengembang Amazon Q

Amazon Q Developer tersedia sebagai bagian dari AWS Toolkit for Visual Studio Code dan AWS Toolkit for JetBrains (misalnya, PyCharm IntelliJ dan). Untuk Visual Studio Code dan JetBrains IDEs, Amazon Q Developer mendukung Python, Java,, C # JavaScript, Go TypeScript, Rust, PHP, Ruby, Kotlin, C, C ++, Shell scripting, SQL, dan Scala.

Untuk petunjuk mendetail tentang cara menginstal AWS Toolkit untuk Visual Studio Code dan JetBrains IDE, lihat [Menginstal ekstensi atau plugin Amazon Q Developer di IDE Anda](#) di Panduan Pengguna Pengembang Amazon Q.

## Praktik terbaik saat menggunakan Amazon Q Developer

Praktik terbaik umum saat menggunakan Amazon Q Developer mencakup hal-hal berikut:

- Berikan konteks yang relevan untuk mendapatkan tanggapan yang lebih akurat, seperti bahasa pemrograman, kerangka kerja, dan alat yang sedang digunakan. Memecah masalah kompleks menjadi komponen yang lebih kecil.
- Eksperimen dan ulangi permintaan dan pertanyaan Anda. Pemrograman sering melibatkan mencoba pendekatan yang berbeda.

- Selalu tinjau saran kode sebelum Anda menerimanya, dan edit sesuai kebutuhan untuk memastikan bahwa mereka melakukan persis seperti yang Anda inginkan.
- Gunakan [kemampuan penyesuaian](#) untuk membuat Pengembang Amazon Q mengetahui pustaka internal, praktik terbaik APIs, dan pola arsitektur Anda untuk rekomendasi yang lebih relevan.

## Privasi data dan penggunaan konten di Amazon Q Developer

Saat memutuskan untuk menggunakan Amazon Q Developer, Anda harus memahami bagaimana data dan konten Anda digunakan. Berikut ini adalah poin-poin penting:

- Untuk pengguna Amazon Q Developer Pro, konten kode Anda tidak digunakan untuk peningkatan layanan atau pelatihan model.
- Untuk pengguna Tingkat Gratis Pengembang Amazon Q, Anda dapat memilih untuk tidak menggunakan konten Anda untuk peningkatan layanan melalui pengaturan atau AWS Organizations kebijakan IDE.
- Konten yang ditransmisikan dienkripsi, dan setiap konten yang disimpan diamankan dengan enkripsi saat istirahat dan kontrol akses. Untuk informasi selengkapnya, lihat [Enkripsi data di Amazon Q Developer](#) di Panduan Pengguna Pengembang Amazon Q.

## Praktik terbaik untuk pembuatan kode dengan Amazon Q Developer

Amazon Q Developer menyediakan pembuatan kode otomatis, pelengkapan otomatis, dan saran kode bahasa alami. Berikut ini adalah praktik terbaik untuk menggunakan bantuan pengkodean in-line Pengembang Amazon Q:

- Berikan konteks untuk membantu meningkatkan akurasi tanggapan

Mulailah dengan kode yang ada, impor pustaka, buat kelas dan fungsi, atau buat kerangka kode. Konteks ini akan membantu meningkatkan kualitas pembuatan kode secara signifikan.

- Kode secara alami

Gunakan pembuatan kode Pengembang Amazon Q seperti mesin pelengkapan otomatis yang kuat. Kode seperti yang biasa Anda lakukan, dan biarkan Amazon Q memberikan saran saat Anda mengetik atau menjeda. Jika pembuatan kode tidak tersedia atau Anda terjebak pada masalah kode, lakukan Amazon Q dengan mengetikkan Alt+C di PC atau Option+C di macOS. Untuk

informasi selengkapnya tentang tindakan umum yang dapat Anda lakukan saat menggunakan saran in-line, lihat [Menggunakan tombol pintas](#) di Panduan Pengguna Pengembang Amazon Q.

- Sertakan pustaka impor yang relevan dengan tujuan skrip Anda

Sertakan pustaka impor yang relevan untuk membantu Amazon Q memahami konteksnya dan menghasilkan kode yang sesuai. Anda juga dapat meminta Amazon Q untuk menyarankan pernyataan impor yang relevan.

- Pertahankan konteks yang jelas dan fokus

Jaga agar skrip Anda tetap fokus pada tujuan tertentu, dan modulasi fungsionalitas yang berbeda menjadi skrip terpisah dengan konteks yang relevan. Hindari konteks yang bising atau membingungkan.

- Bereksperimenlah dengan petunjuk

Jelajahi berbagai petunjuk untuk mendorong Amazon Q untuk menghasilkan hasil yang berguna dalam pembuatan kode. Misalnya, coba pendekatan berikut:

- Gunakan blok komentar standar untuk permintaan bahasa alami.
  - Buat kerangka dengan komentar untuk mengisi kelas dan fungsi.
  - Jadilah spesifik dalam petunjuk Anda, berikan detail alih-alih generalisasi.
- Mengobrol dengan Pengembang Amazon Q dan minta bantuan

Jika Pengembang Amazon Q tidak memberikan saran yang akurat, ngobrol dengan Pengembang Amazon Q di IDE Anda. Ini dapat memberikan cuplikan kode atau kelas dan fungsi lengkap untuk memulai konteks Anda. Untuk informasi selengkapnya, lihat [Mengobrol dengan Pengembang Amazon Q tentang kode](#) di Panduan Pengguna Pengembang Amazon Q.

## Praktik terbaik untuk rekomendasi kode dengan Amazon Q Developer

Amazon Q Developer dapat mengajukan pertanyaan pengembang dan mengevaluasi kode untuk menawarkan rekomendasi yang berkisar dari pembuatan kode dan perbaikan bug hingga panduan menggunakan bahasa alami. Berikut ini adalah praktik terbaik untuk menggunakan obrolan di Amazon Q:

- Hasilkan kode dari awal

Untuk proyek baru atau ketika Anda memerlukan fungsi umum (misalnya, menyalin file dari Amazon S3), minta Pengembang Amazon Q untuk menghasilkan contoh kode menggunakan petunjuk bahasa alami. Amazon Q dapat menyediakan tautan terkait ke sumber daya publik untuk validasi dan penyelidikan lebih lanjut.

- Cari pengetahuan pengkodean dan penjelasan kesalahan

Saat dihadapkan dengan masalah pengkodean atau pesan kesalahan, berikan blok kode (dengan pesan kesalahan, jika ada) dan pertanyaan Anda sebagai prompt ke Pengembang Amazon Q. Konteks ini akan membantu Amazon Q memberikan tanggapan yang akurat dan relevan.

- Tingkatkan kode yang ada

Untuk memperbaiki kesalahan yang diketahui atau mengoptimalkan kode (misalnya, untuk mengurangi kompleksitas), pilih blok kode yang relevan dan kirimkan ke Pengembang Amazon Q dengan permintaan Anda. Jadilah spesifik dengan petunjuk Anda untuk hasil yang lebih baik.

- Jelaskan fungsionalitas kode

Saat menjelajahi repositori kode baru, pilih blok kode atau seluruh skrip dan kirimkan ke Pengembang Amazon Q untuk penjelasan. Kurangi ukuran seleksi untuk penjelasan yang lebih spesifik.

- Hasilkan pengujian unit

Setelah mengirim blok kode sebagai prompt, minta Pengembang Amazon Q untuk menghasilkan pengujian unit. Pendekatan ini dapat menghemat waktu dan biaya pengembangan yang terkait dengan cakupan kode dan DevOps.

- Temukan AWS jawaban

Amazon Q Developer adalah sumber daya berharga bagi pengembang yang bekerja dengan Layanan AWS karena mengandung sejumlah besar pengetahuan yang terkait dengannya AWS. Baik Anda menghadapi tantangan dengan pesan kesalahan tertentu yang spesifik Layanan AWS, atau mencoba mempelajari yang baru AWS, Layanan AWS Amazon Q sering memberikan informasi yang relevan dan berguna.

Selalu tinjau rekomendasi yang diberikan Amazon Q Developer kepada Anda. Kemudian, lakukan pengeditan yang diperlukan dan jalankan tes untuk memastikan bahwa kode tersebut memenuhi fungsionalitas yang Anda inginkan.

# Contoh kode dengan Pengembang Amazon Q

Bagian ini memberikan contoh realistis tentang cara meningkatkan pengalaman dan pembuatan kode Anda dengan fokus pada Python and Java bahasa. Seiring dengan contoh in-line, beberapa skenario menunjukkan bagaimana Anda dapat menggunakan kemampuan obrolan Amazon Q Developer untuk meminta petunjuk rekomendasi kode saat Anda memprogram.

Untuk informasi selengkapnya, lihat topik berikut:

- [Python contoh](#)
- [Java contoh](#)
- [Contoh obrolan](#)

## Python contoh

Berikut ini Python contoh menunjukkan bagaimana Anda dapat memulai membuat saran kode in-line dengan Amazon Q Developer. Contoh ilustrasi ini menunjukkan kepada Anda cara memicu asisten pengkodean untuk proyek Anda sendiri. Sebagian besar contoh berikut berlaku untuk bahasa lain:

- [Menghasilkan kelas dan fungsi](#)
- [Kode dokumen](#)
- [Menghasilkan algoritma](#)
- [Hasilkan pengujian unit](#)

## Menghasilkan kelas dan fungsi

Untuk menghasilkan implementasi kelas penuh atau sebagian, gunakan komentar kode. Komentar tersebut menjelaskan maksud kelas. Mulai sederhana, dan tambahkan detail lebih lanjut jika perlu. Setelah Anda menghasilkan kelas, Anda dapat terus menghasilkan fungsi dalam kerangkanya.

Saat Anda menulis komentar, lebih baik menggunakan struktur komentar bahasa. Masuk Python, untuk komentar baris tunggal, gunakan `#`. Untuk komentar multiline, gunakan `''' '''` atau `""" """`.

Masukan:

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

Output:

```
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
```

Sekarang ada kelas awal, fungsi tambahan dapat dihasilkan dengan komentar tambahan.

Masukan:

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

Output:

```
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}")
```

## Kode dokumen

Kurangi waktu dalam dokumentasi kode dengan mengandalkan dokumentasi in-line. Tambahkan Python jeda komentar ''' '''.

Masukan:

```
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}")
```

Output:

```
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}")
```

## Menghasilkan algoritma

Amazon Q Developer dapat menghasilkan algoritma populer, terutama yang terkait AWS SDKs dengan. Dalam skrip Anda, tambahkan prompt yang menentukan fungsi yang Anda inginkan.

Contoh ini menunjukkan cara menghasilkan kamus kunci id pengguna untuk memberi nama nilai, dan kemudian menghasilkan fungsi penyortiran dengan menggunakan algoritma pengurutan gelembung.

Masukan:

```
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
...
```

Output:

```
@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)
```

Selain itu, Anda dapat membuat petunjuk untuk prosedur standar. Dalam contoh ini, pengguna membuat prompt untuk menyimpan kamus ke tabel Amazon DynamoDB. Di awal skrip, pengguna menambahkan prompt untuk menambahkan pustaka DynamoDB untuk membantu menyediakan Amazon Q Developer dengan konteks tambahan.

Masukan:

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

Output:

```
'''
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')

```

Sekarang, pengguna dapat dengan mudah meminta Amazon Q untuk membantu menyimpan kamus di DynamoDB.

Masukan:

```

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

```

Output:

```

@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'])

```

## Hasilkan pengujian unit

Dengan komentar in-line, Pengembang Amazon Q dapat menghasilkan pengujian unit untuk melakukan pernyataan dan pengujian lainnya. Dalam contoh ini, pengguna meminta Amazon

Q untuk menghasilkan pustaka yang berlaku untuk pengujian unit guna menyetel konteks skrip pengujian. Kemudian, pengguna meminta Amazon Q untuk menulis fungsi pengujian untuk `sort_user_id_name_map` fungsi tersebut.

Masukan:

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

Output:

```
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()
```

Selain itu, dalam obrolan antara pengguna dan Pengembang Amazon Q, ini menyarankan dan menghasilkan pengujian unit berdasarkan permintaan input kode pengguna. Untuk informasi selengkapnya, lihat [Contoh obrolan](#).

## Java contoh

Berikut ini Java contoh menunjukkan bagaimana pengguna dapat memulai dalam menghasilkan saran kode in-line. Contoh ilustrasi ini menunjukkan kepada Anda cara memicu asisten pengkodean untuk proyek Anda sendiri. Sebagian besar contoh berikut berlaku untuk bahasa lain:

- [Menghasilkan kelas dan fungsi](#)
- [Kode dokumen](#)
- [Menghasilkan algoritma](#)

- [Hasilkan pengujian unit](#)

## Menghasilkan kelas dan fungsi

Untuk menghasilkan implementasi kelas penuh atau sebagian, gunakan komentar kode. Komentar tersebut menjelaskan maksud kelas. Mulai sederhana, dan tambahkan detail lebih lanjut jika perlu. Setelah Anda menghasilkan kelas, Anda dapat terus menghasilkan fungsi dalam kerangkanya.

Saat menulis komentar, lebih baik menggunakan struktur komentar bahasa. Masuk Java, ini adalah `/** */`.

Masukan:

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

Output:

```
/**
 * 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);
    }
}

```

Sekarang ada kelas awal, fungsi tambahan dapat dihasilkan dengan komentar tambahan.

Masukan:

```

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

```

Output:

```

/**
 * 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());
    }
}

```

## Kode dokumen

Kurangi waktu dalam dokumentasi kode dengan mengandalkan dokumentasi in-line. Tambahkan Java istirahat komentar `/** */`.

Masukan:

```
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;
        };
    }
}
```

Jika pembuatan in-line tidak terjadi, jalankan saran in-line Amazon Q Developer setelah menempatkan kursor di tempat yang Anda inginkan dokumentasi.

Output:

```
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;
        };
    }
}
```

Untuk penjelasan kode yang lebih sederhana (kelas, fungsi, atau elemen skrip), gunakan karakter maju (//) yang ditempatkan secara strategis sebelum bagian kode tempat Anda ingin menambahkan dokumentasi.

Masukan:

```
//
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;
        };
    }
}
```

Output:

```
// 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;
        };
    }
}
```

## Menghasilkan algoritma

Amazon Q Developer dapat menghasilkan algoritma populer, terutama yang terkait AWS SDKs dengan. Dalam skrip Anda, tambahkan prompt yang menentukan fungsi yang Anda inginkan.

Contoh ini menghasilkan peta hash kunci id pengguna untuk memberi nama nilai dan kemudian menghasilkan fungsi penyortiran menggunakan algoritma pengurutan gelembung.

Masukan:

```
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
    */

}
```

Output:

```
/*
* 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;  
}
```

Selain itu, pengguna dapat membuat petunjuk untuk prosedur standar. Dalam contoh ini, pengguna membuat prompt untuk menyimpan peta hash ke tabel Amazon DynamoDB. Di awal skrip, pengguna menambahkan prompt untuk menambahkan DynamoDB Java library untuk membantu menyediakan Amazon Q Developer dengan konteks tambahan.

Masukan:

```
// 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  
 */
```

Output:

```
// 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);  
        }});  
    }  
}
```

```
        }));  
    }  
}
```

## Hasilkan pengujian unit

Dengan komentar in-line, Pengembang Amazon Q dapat menghasilkan pengujian unit untuk melakukan pernyataan dan pengujian lainnya. JUnit Dalam contoh ini, pengguna pertama-tama meminta Amazon Q untuk membuat pustaka yang berlaku JUnit untuk mengatur konteks skrip pengujian. Selanjutnya, pengguna membuat kelas publik diikuti oleh deklarator dan string komentar.

Masukan:

```
// import libraries for java unit tests  
import org.junit.Test;  
import static org.junit.Assert.*;  
  
public class UnitTests {  
  
    @Test  
    /*  
     * Create unit test for getUserIdNameMap  
     */  
}
```

Output:

```
@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);
}
```

Selain itu, dalam obrolan antara pengguna dan Pengembang Amazon Q, ini menyarankan dan menghasilkan pengujian unit berdasarkan permintaan input kode pengguna. Untuk informasi selengkapnya, lihat [Contoh obrolan](#).

## Contoh obrolan

Obrolan Amazon Q Developer membantu pengembang dengan berbagai tugas mulai dari pengkodean, pengujian, dan peningkatan aplikasi, hingga mendiagnosis kesalahan, melakukan pemindaian dan perbaikan keamanan, dan mengoptimalkan sumber daya. AWS Bagian ini membahas bagaimana mengobrol dengan Amazon Q dapat mendukung pengembang dalam tugas pemrograman khas seperti berikut:

- [Tanyakan tentang Layanan AWS](#)
- [Hasilkan kode](#)
- [Hasilkan pengujian unit](#)
- [Jelaskan kode](#)

## Tanyakan tentang Layanan AWS

Pengguna dapat mengajukan pertanyaan tentang Amazon Q Developer Layanan AWS, seperti menemukan layanan yang tepat atau memahami praktik terbaik.

Mengobrol tentang Layanan AWS

Pengguna:

Apa layanan kontainer terbaik yang digunakan untuk menjalankan beban kerja saya jika saya perlu menjaga biaya saya tetap rendah?

Amazon Q:

Berdasarkan informasi yang diberikan, sepertinya Anda mencari layanan kontainer yang hemat biaya untuk menjalankan beban kerja Anda. Dalam hal ini, saya akan merekomendasikan untuk mempertimbangkan Amazon Elastic Container Service (Amazon ECS) dengan Fargate. [1]

Amazon ECS dengan Fargate adalah mesin komputasi tanpa server untuk kontainer yang memungkinkan Anda menjalankan kontainer tanpa harus mengelola infrastruktur yang mendasarinya. Dengan Fargate, Anda tidak perlu menyediakan atau mengelola server — Fargate mengurusnya untuk Anda. Ini dapat membantu menjaga biaya tetap rendah karena Anda hanya membayar sumber daya komputasi yang benar-benar digunakan kontainer Anda, tanpa harus membayar kapasitas server idle. [2]

Beberapa manfaat utama menggunakan Amazon ECS dengan Fargate:

- Tidak perlu mengelola server atau cluster - Fargate menangani penyediaan dan manajemen infrastruktur untuk Anda.
- Pay-as-you-go harga — Anda hanya membayar sumber daya komputasi yang digunakan kontainer Anda, tanpa harus membayar kapasitas idle.
- Penskalaan otomatis - Fargate dapat secara otomatis meningkatkan dan menurunkan kontainer Anda berdasarkan permintaan.
- Integrasi ketat dengan lainnya Layanan AWS seperti load balancing, logging, dan monitoring.

Sumber daya yang disediakan juga menyebutkan bahwa layanan ECS menawarkan fleksibilitas dalam hal penjadwalan dan penempatan kontainer, yang dapat membantu mengoptimalkan biaya Anda lebih lanjut.

Saya akan merekomendasikan meninjau detail harga untuk Amazon ECS dengan Fargate di AWS situs web untuk mendapatkan pemahaman yang lebih baik tentang biaya berdasarkan persyaratan beban kerja spesifik Anda. Dan tentu saja, jangan ragu untuk memberi tahu saya jika Anda memiliki pertanyaan lain!

Sumber

[1] Layanan AWS Kontainer Mana yang Terbaik untuk Anda?

[aptgetops.tech > - -Anda aws-container-services-which one-is-right-for](#)

[2] Java opsi kontainer untuk workload non-HTTP

[stackoverflow.com > pertanyaan > 43417016](#)

## Akhiri obrolan Amazon Q Layanan AWS

### Hasilkan kode

Pengguna dapat meminta bantuan Pengembang Amazon Q untuk menulis cuplikan kode.

Mengobrol tentang menulis kode

Pengguna:

Bisakah Anda memberi saya contoh cara membuat daftar semua ember S3 menggunakan? AWS SDK for Python (Boto)

Amazon Q:

Tentu saja, berikut adalah contoh cara membuat daftar semua bucket S3 menggunakan AWS SDK for 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'])
```

Begini cara kerja kode:

1. Kami pertama kali mengimpor perpustakaan boto3, yang merupakan SDK untuk Python.
2. Kami membuat klien S3 menggunakan `boto3.client('s3')`.
3. Kami memanggil `list_buckets()` metode pada klien S3, yang mengembalikan respons yang berisi daftar semua ember di akun.
4. Kami kemudian mengulangi Buckets daftar dalam respons dan mencetak nama setiap ember.

Ini adalah contoh dasar, tetapi AWS SDK for Python (Boto) menyediakan lebih banyak fitur canggih untuk bekerja dengan Amazon S3, seperti membuat, menghapus, dan mengelola objek dalam

bucket, menetapkan kebijakan bucket, dan banyak lagi. Anda dapat menemukan lebih banyak contoh dan dokumentasi di dokumentasi AWS SDK for Python (Boto)). [2]

Ingat, ketika bekerja dengan Layanan AWS, penting untuk mengikuti praktik terbaik untuk menangani kredensial dan informasi sensitif, seperti menyimpan kunci AWS akses dalam variabel lingkungan alih-alih mengkodekannya dalam kode.

Sumber: [1] Contoh [Amazon S3 menggunakan SDK untuk Python \( AWS Boto3\) - Contoh Kode SDK docs.aws.amazon.com > pustaka kode > terbaru-1x >... > python-s3.html](https://docs.aws.amazon.com/sdk-for-python/v2/docs/example_s3_buckets.html)

[2] [Memulai dengan bucket dan objek Amazon S3 menggunakan AWS SDK - Contoh Kode SDK docs.aws.amazon.com> code-library > latest-1x >... AWS > s3\\_example\\_s3\\_buckets\\_ \\_sect... GettingStarted](https://docs.aws.amazon.com/sdk-for-python/v2/docs/example_s3_buckets.html)

Akhiri obrolan Amazon Q tentang menulis kode

## Hasilkan pengujian unit

Pengguna dapat mengirim kode yang dipilih sebagai prompt ke Amazon Q Developer, di mana instruksi tambahan diterapkan seperti membuat pengujian unit. Untuk step-by-step petunjuk, lihat [Mengirim kode ke Amazon Q](#) di Panduan Pengguna Pengembang Amazon Q.

Mengobrol tentang pengujian unit

Pengguna:

Tulis unit test dengan data tiruan untuk fungsi berikut:

```
// 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:

Untuk menulis unit test dengan data tiruan untuk fungsi `pollMessages`, kita dapat menggunakan kerangka kerja mengejek seperti Mockito. Inilah contohnya:

```
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);
}
}
```

Dalam contoh ini, kami menggunakan Mockito untuk mengejek klien AmazonSQS dan mengatur perilaku yang diharapkan untuk metode `receiveMessage` sehingga mengembalikan dua pesan tiruan dengan badan "message1" dan "message2". Kemudian, kami memanggil metode `pollMessages` dan menegaskan bahwa daftar pesan yang dikembalikan cocok dengan nilai yang diharapkan.

Poin kuncinya adalah:

1. Kami menggunakan anotasi `@ ExtendWith (MockitoExtension.class)` untuk mengaktifkan integrasi Mockito dengan JUnit
2. Kami menggunakan anotasi `@Mock` untuk membuat instance tiruan dari klien AmazonSQS.
3. Kami menggunakan `InjectMocks` anotasi `@` untuk menyuntikkan klien AmazonSQS tiruan ke dalam kelas. `SQSUtils`
4. Dalam metode pengujian, kita mengatur perilaku yang diharapkan dari metode `receiveMessage` menggunakan `mockito.When () .thenReturn ()`.
5. Kami memanggil metode `pollMessages` dan menegaskan bahwa daftar pesan yang dikembalikan cocok dengan nilai yang diharapkan.

Pendekatan ini memungkinkan kami untuk menguji metode `PollMessages` tanpa benar-benar berinteraksi dengan layanan Amazon SQS, yang dapat membuat pengujian lebih cepat dan lebih andal.

Akhiri obrolan Amazon Q tentang pengujian unit

## Jelaskan kode

Untuk proses peninjauan kode, pengembang dapat mengirim pilihan kode ke Pengembang Amazon Q untuk lebih memahami kode. Selain itu, penjelasan yang disediakan Amazon Q juga dapat melayani tujuan dokumentasi. Dalam contoh berikut, Amazon Q menjelaskan fungsi handler dengan AWS Lambda fungsi, dan kemudian mengusulkan docstring.

Mengobrol tentang menjelaskan kode

Pengguna:

Jelaskan bagian berikut dari kode saya:

```
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:

Blok kode yang disediakan tampaknya merupakan AWS Lambda fungsi yang ditulis dalam Python. Izinkan saya menjelaskan aspek-aspek kunci dari kode ini:

1. Fungsi ini bernama `lambda_handler` dan itu adalah titik masuk untuk fungsi Lambda.
2. Ini menggunakan pustaka `boto3` untuk berinteraksi dengan layanan AWS DynamoDB dan S3.
3. Fungsi mengambil kunci objek S3 dan nama bucket dari data peristiwa yang masuk.

4. Kemudian memanggil fungsi `detect_text` (yang tidak ditampilkan dalam kode yang disediakan) untuk mengekstrak konten teks lengkap dari objek S3.
5. Fungsi ini juga mengambil metadata tentang objek S3 menggunakan metode `head_object`.
6. Kemudian mengekstrak waktu saat ini dan memanggil fungsi `detect_restaurant_closure` (juga tidak ditampilkan) untuk menentukan alasan dan hasil penutupan restoran berdasarkan teks yang diekstraksi.
7. Akhirnya, fungsi menyimpan informasi berikut dalam tabel DynamoDB:
  - Kunci objek S3 sebagai id
  - Waktu penciptaan
  - Alasan dan hasil penutupan restoran
  - Teks objek S3 yang diekstraksi

Secara keseluruhan, fungsi Lambda ini tampaknya menjadi bagian dari sistem yang lebih besar yang memproses objek S3, mengekstrak konten teks, menganalisisnya untuk informasi penutupan restoran, dan menyimpan hasilnya dalam tabel DynamoDB. Detail spesifik dari fungsi `detect_text` dan `detect_restaurant_closure` akan diperlukan untuk sepenuhnya memahami fungsionalitas kode ini.

Akhir dari respons Amazon Q

Pengguna: rangkum penjelasan ini menjadi docstring untuk fungsi `lambda_handler` ini

Amazon Q: Berikut adalah docstring yang disarankan untuk fungsi `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...
```



# Memecahkan masalah skenario pembuatan kode di Amazon Q Developer

Anda mungkin mengalami skenario umum berikut dengan pembuatan dan resolusi kode yang tidak akurat saat menggunakan Amazon Q Developer:

- [Pembuatan kode kosong](#)
- [Komentar terus menerus](#)
- [Pembuatan kode in-line salah](#)
- [Hasil yang tidak memadai dari obrolan](#)

## Pembuatan kode kosong

Saat mengembangkan kode, Anda mungkin melihat masalah berikut:

- Amazon Q tidak memberikan saran.
- Pesan “Tidak Ada Saran dari Amazon Q” muncul di IDE Anda.

Pikiran pertama Anda mungkin bahwa Amazon Q tidak berfungsi dengan baik. Namun, akar penyebab masalah ini biasanya terkait dengan konteks dalam skrip atau proyek terbuka dalam IDE.

Jika Pengembang Amazon Q tidak memberikan saran secara otomatis, Anda dapat menggunakan pintasan berikut untuk menjalankan saran in-line Amazon Q secara manual:

- PC - Alt+C
- macOS - Opsi+C

Untuk informasi selengkapnya, lihat [Menggunakan tombol pintas](#) di Panduan Pengguna Pengembang Amazon Q.

Dalam sebagian besar skenario, Amazon Q menghasilkan saran. Saat Amazon Q mengembalikan pesan “Tidak Ada Saran dari Amazon Q”, tinjau resolusi umum berikut untuk masalah ini:

- Konteks yang tidak memadai — Pastikan pustaka yang berlaku ada. Pastikan bahwa kelas dan fungsi yang didefinisikan dalam skrip berhubungan dengan kode baru.

- Prompt ambigu — Jika prompt membingungkan atau tidak jelas, Amazon Q akan berkinerja buruk saat membuat saran kode in-line. Dalam skenario ini, lebih spesifik dalam kode dan komentar Anda untuk mendapatkan tanggapan yang tepat.

Jika Anda terjebak, lanjutkan pengkodean dengan kemampuan terbaik Anda. Biasanya, Amazon Q akan mengenali niat Anda, yang memicunya untuk menawarkan saran. Pilihan lain adalah mengirim cuplikan kode dengan instruksi cepat dalam obrolan Amazon Q. Untuk informasi selengkapnya, lihat [Mengirim kode ke Amazon Q](#) di Panduan Pengguna Pengembang Amazon Q.

## Komentar terus menerus

Anda dapat menulis prompt menggunakan komentar dan hanya menerima baris komentar umum atau berulang dari Amazon Q.

Masukan:

```
// Kafka test
```

Hasil:

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

Biasanya, masalah ini disebabkan oleh kurangnya konteks dalam skrip atau instruksi prompt yang buruk. Kami menyarankan Anda menggunakan format standar untuk komentar yang diberikan bahasa. Dalam contoh ini, pengguna mengubah prompt yang akan digunakan `/* */` untuk blok komentar, dan perubahan itu menghasilkan kinerja yang lebih baik dari Amazon Q.

Masukan:

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

Hasil:

Dalam hasil ini, pengguna menerima beberapa saran secara berurutan.

```
/*
 * 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());
    }
}
```

## Pembuatan kode in-line salah

Terkadang, saran in-line Amazon Q Developer tidak berguna atau tidak memenuhi tujuan prompt Anda. Kadang-kadang, saran kode in-line mungkin menyediakan fungsi generik yang mirip dengan metode Hello World sederhana.

Jika saran kode in-line tidak membantu, coba praktik terbaik berikut:

- Tinjau konteks naskah. Biasanya, semakin banyak konteks yang Anda berikan, semakin baik hasil yang Anda dapatkan dari Amazon Q.

- Jika konteksnya bervariasi and/or berisik, modularisasi kode Anda. Kelompokkan fungsi serupa dalam skrip yang sama. Pisahkan kelas dan fungsi yang berbeda berdasarkan tujuannya.
- Tulis kelas dan nama fungsi Anda secara taktis untuk membantu Amazon Q memberikan rekomendasi yang lebih baik.

Dalam contoh ini, pengembang menulis skrip dari awal yang mengunggah file ke Hadoop.

Masukan:

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

Hasil:

```
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));  
  
    }  
}
```

Meskipun Amazon Q memberikan rekomendasi, itu tidak memenuhi harapan pengembang. Untuk meningkatkan kinerja, pengembang mengimpor pustaka dan mengubah nama kelas untuk mewakili utilitas Hadoop.

Perbaikan:

```
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  
}
```

Hasil:

Dalam hasil ini, pengembang menerima beberapa saran secara berurutan.

```
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!");  
    }  
}
```

Dalam HadoopUtilities skrip yang sama, pengembang menambahkan prompt komentar untuk membaca topik Amazon Simple Queue Service (Amazon SQS).

Masukan:

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

Hasil:

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

Alih-alih menggunakan metode Java untuk membaca pesan SQS dari Amazon SQS, Amazon Q menyarankan untuk mencetak metode string. Namun, saran ini tidak memenuhi tujuan awal pengembang.

Dalam hal ini, lebih baik untuk memodulasi ulang kode. Fokus skrip adalah pada metode Hadoop. Oleh karena itu, mungkin pendekatan yang lebih baik untuk membuat skrip utilitas yang khusus untuk operasi SQS.

Membuat kelas terpisah untuk utilitas SQS meningkatkan saran in-line, yang dapat digunakan dalam skrip. HadoopUtilities

Masukan:

```
// 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
}
```

Hasil:

```
// 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;
```

Jika saran in-line masih belum memberikan hasil yang diharapkan, minta Pengembang Amazon Q dalam sesi obrolan untuk menghasilkan kode.

## Mengobrol tentang menghasilkan kode

Pengguna:

Menghasilkan fungsi java untuk membaca pesan dari antrian Amazon SQS

Amazon Q:

Berikut adalah contoh fungsi Java untuk membaca pesan dari antrian 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;
}
}
```

Kode ini menciptakan kelas SQSMessage Reader yang mengambil klien AmazonSQS dan URL antrian sebagai parameter konstruktor. Metode readMessage () menggunakan ReceiveMessageRequest untuk mengambil hingga 1 pesan dari antrian, dengan waktu tunggu maksimum 10 detik.

Jika pesan diterima, badan pesan dicetak dan pesan dihapus dari antrian menggunakan DeleteMessageRequest Badan pesan kemudian dikembalikan. Jika tidak ada pesan yang diterima, null dikembalikan.

Akhiri obrolan Amazon Q tentang menghasilkan kode

## Hasil yang tidak memadai dari obrolan

Saat mengembangkan kode, pengembang dapat meminta Amazon Q untuk menghasilkan fungsi umum, memberikan rekomendasi, atau menjelaskan kode. Terkadang, dengan prompt (seperti pertanyaan atau cuplikan kode), Amazon Q dapat memberikan umpan balik umum atau hasil yang tidak memenuhi harapan Anda. Dalam skenario ini, coba yang berikut ini:

- Bereksperimenlah dengan berbagai petunjuk untuk meningkatkan hasil pembuatan kode yang Anda dapatkan dari Amazon Q.
- Ikuti URLs yang sering diberikan Amazon Q dalam tanggapannya terhadap permintaan atau pertanyaan Anda.
- Berikan umpan balik untuk membantu meningkatkan kinerja Amazon Q di masa mendatang.

Misalnya, pengembang membuat solusi Retrieval Augmented Generation (RAG) dan mereka perlu memotong dokumen mereka mengingat hubungan semantik antar kalimat. Pengembang mengobrol dengan Amazon Q sebagai berikut.

## Mengobrol tentang solusi RAG

Pengguna:

Untuk solusi RAG, buat kode untuk chunking semantik

Amazon Q:

Tentu saja, berikut adalah contoh bagaimana Anda dapat menerapkan chunking semantik untuk solusi Retrieval Augmented Generation (RAG) dalam menggunakan perpustakaan: Python 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
```

Dalam contoh ini, kita menggunakan `TokenTextSplitter` from `LangChain` untuk membagi teks input menjadi potongan-potongan 500 karakter dengan tumpang tindih 50 karakter. `HuggingFaceEmbeddingsModel` ini digunakan untuk menghasilkan embeddings untuk setiap

potongan, dan penyimpanan Chroma vektor digunakan untuk menyimpan potongan dan penyematannya.

Fungsi `semantic_chunk` mengambil teks input, membaginya menjadi potongan-potongan menggunakan `text_splitter`, menghasilkan embeddings untuk setiap potongan menggunakan model embeddings, dan kemudian menambahkan potongan dan embeddings mereka ke `vectorstore`

Pendekatan ini memungkinkan Anda untuk membagi teks input menjadi potongan-potongan yang bermakna secara semantik, yang dapat berguna untuk berbagai tugas, seperti menjawab pertanyaan atau meringkas, dalam solusi RAG.

Harap dicatat bahwa ini adalah contoh dasar, dan Anda mungkin perlu menyesuaikan parameter (misalnya, `chunk_size`, `chunk_overlap`) atau pilihan model splitter dan embeddings teks berdasarkan kasus penggunaan dan data spesifik Anda.

Akhir obrolan Amazon Q tentang solusi RAG

Namun, pendekatan ini hanya menyediakan cara dasar untuk memotong dokumen dengan menggunakan teknik jendela geser. Pengembang menginginkan cara canggih yang bergantung pada pengukuran embeddings. Dalam skenario ini, pengembang harus menambahkan lebih spesifik ke prompt mengingat tujuannya. Ketika pengembang mengacu pada fungsi splitter AI21 teks yang dikenal dengan LangChain, mereka menerima respons yang lebih baik seperti yang ditunjukkan dalam obrolan berikut.

### Mengobrol tentang solusi RAG (dengan prompt yang ditingkatkan)

Pengguna:

Prompt di atas hanya berkaitan dengan pemisahan mengenai ukuran token per potongan. Saya ingin menggunakan AI21 pembagi teks Semantik. Hasilkan fungsi menggunakan pemisahan teks semantik AI21 dengan langchain.

Amazon Q:

Oke, mari kita gunakan AI21 Semantic Text Splitter LangChain untuk mengimplementasikan chunking semantik. Berikut adalah contoh: [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
```

Dalam contoh ini, kita menggunakan AI21 SemanticTextSplitter from LangChain untuk membagi teks input menjadi potongan-potongan yang bermakna secara semantik. HuggingFaceEmbeddingsModel ini digunakan untuk membuat embeddings untuk setiap potongan, dan penyimpanan Chroma vektor digunakan untuk menyimpan potongan dan penyematannya.

Fungsi semantic\_chunk mengambil teks input, membaginya menjadi potongan-potongan menggunakan text\_splitter, membuat embeddings untuk setiap potongan menggunakan model embeddings, dan kemudian menambahkan potongan dan embeddings mereka ke vectorstore

Pendekatan ini memungkinkan Anda untuk membagi teks input menjadi potongan-potongan yang bermakna secara semantik, yang dapat berguna untuk berbagai tugas, seperti menjawab pertanyaan atau meringkas, dalam solusi RAG.

Harap dicatat bahwa Anda mungkin perlu menyesuaikan parameter atau pilihan model embeddings berdasarkan kasus penggunaan dan data spesifik Anda.

Sumber [1] Pelajari cara menggunakan pemisah teks di LangChain

[soosweb3.hashnode.dev > -pemisah teks the-ultimate-langchain-series](https://soosweb3.hashnode.dev/-pemisah-teks-the-ultimate-langchain-series)

Akhir obrolan Amazon Q tentang solusi RAG (dengan prompt yang ditingkatkan)

Berdasarkan hasil obrolan, pengembang dapat menggunakan fungsi tersebut dan menavigasi ke sumber informasi.

Secara umum, topik dan pertanyaan lanjutan memerlukan konteks yang lebih spesifik dalam prompt saat Anda mengobrol dengan Amazon Q Developer. Jika Anda yakin bahwa hasil dari obrolan Anda tidak akurat, gunakan ikon jempol ke bawah untuk memberikan umpan balik tentang respons Amazon Q. Pengembang Amazon Q terus menggunakan umpan balik untuk meningkatkan rilis future. Untuk interaksi yang menghasilkan hasil positif, ada baiknya memberikan umpan balik Anda dengan menggunakan ikon acungan jempol.

## FAQs tentang Pengembang Amazon Q

Bagian ini memberikan jawaban atas pertanyaan umum tentang penggunaan Amazon Q Developer untuk pengembangan kode.

### Apa itu Pengembang Amazon Q?

Amazon Q Developer adalah layanan bertenaga AI generatif yang kuat yang dirancang untuk mempercepat tugas pengembangan kode dengan menyediakan pembuatan dan rekomendasi kode cerdas. Pada 30 April 2024, Amazon CodeWhisperer menjadi bagian dari Pengembang Amazon Q.

### Bagaimana cara mengakses Amazon Q Developer?

Amazon Q Developer tersedia sebagai bagian dari AWS Toolkit untuk Visual Studio Code dan JetBrains IDEs, seperti IntelliJ dan PyCharm Untuk memulai, [instal AWS Toolkit versi terbaru](#).

### Bahasa pemrograman apa yang didukung Pengembang Amazon Q?

Untuk Visual Studio Code dan JetBrains IDEs, Amazon Q Developer mendukung Python, Java, C # JavaScript TypeScript, PHP, GoRust, Kotlin, CRuby, C ++, Shell scripting, SQL, dan Scala. Meskipun panduan ini berfokus pada Python dan Java misalnya tujuan, konsep tersebut berlaku untuk bahasa pemrograman yang didukung.

### Bagaimana saya bisa memberikan konteks kepada Pengembang Amazon Q untuk pembuatan kode yang lebih baik?

Mulailah dengan kode yang ada, impor pustaka yang relevan, buat kelas dan fungsi, atau buat kerangka kode. Gunakan blok komentar standar untuk permintaan bahasa alami. Jaga agar skrip Anda tetap fokus pada tujuan tertentu, dan modulasi fungsionalitas yang berbeda menjadi skrip terpisah dengan konteks yang relevan. Untuk informasi selengkapnya, lihat [Praktik pengkodean terbaik dengan Pengembang Amazon Q](#).

## Apa yang harus saya lakukan jika pembuatan kode in-line dengan Pengembang Amazon Q tidak akurat?

Tinjau konteks skrip, pastikan pustaka ada, dan pastikan bahwa kelas dan fungsi berhubungan dengan kode baru. Modularisasi kode Anda, dan pisahkan kelas dan fungsi yang berbeda berdasarkan tujuannya. Tulis petunjuk atau komentar yang jelas dan spesifik. Jika Anda masih tidak yakin tentang keakuratan kode dan Anda tidak dapat melanjutkannya, mulailah obrolan dengan Amazon Q dan kirimkan cuplikan kode dengan instruksi. Untuk informasi selengkapnya, lihat [Memecahkan masalah skenario pembuatan kode di Amazon Q Developer](#).

## Bagaimana cara menggunakan kemampuan obrolan Amazon Q Developer untuk pembuatan kode dan pemecahan masalah?

Mengobrol dengan Amazon Q untuk menghasilkan fungsi umum, meminta rekomendasi, atau menjelaskan kode. Jika respons awal tidak memuaskan, bereksperimenlah dengan petunjuk yang berbeda dan ikuti yang disediakan. URLs Juga, berikan umpan balik ke Amazon Q untuk membantu meningkatkan kinerja obrolan masa depannya. Gunakan ikon jempol ke atas dan jempol ke bawah untuk memberikan umpan balik Anda. Untuk informasi selengkapnya, lihat [Contoh obrolan](#).

## Apa saja praktik terbaik untuk menggunakan Amazon Q Developer?

Berikan konteks yang relevan, eksperimen, dan iterasi pada prompt, tinjau saran kode sebelum menerimanya, menggunakan kemampuan penyesuaian, dan memahami privasi data dan kebijakan penggunaan konten. Untuk informasi selengkapnya, lihat [Praktik terbaik untuk pembuatan kode dengan Amazon Q Developer](#) dan [Praktik terbaik untuk rekomendasi kode dengan Amazon Q Developer](#).

## Dapatkan saya menyesuaikan Pengembang Amazon Q untuk menghasilkan rekomendasi berdasarkan kode saya sendiri?

Ya, gunakan kustomisasi, yang merupakan kemampuan lanjutan dari Pengembang Amazon Q. Dengan penyesuaian, bisnis dapat menyediakan repositori kode mereka sendiri untuk

memungkinkan Pengembang Amazon Q merekomendasikan saran kode in-line. Untuk informasi selengkapnya, lihat [Kemampuan lanjutan Pengembang dan Sumber Daya Amazon Q](#).

# Langkah selanjutnya dalam menggunakan Amazon Q Developer

Dengan pengetahuan yang diperoleh dari panduan komprehensif ini, Anda dapat menggunakan Amazon Q Developer dalam alur kerja pengkodean Anda secara efektif. Instal IDE pilihan Anda ([Visual Studio Code](#) atau [JetBrains](#)), dan mulailah menjelajahi pembuatan kode generatif yang didukung AI dan rekomendasi dari Amazon Q Developer. AWS Toolkit

Cara paling efektif untuk membuka potensi penuh Pengembang Amazon Q adalah dengan pengalaman langsung dengan kode Anda sendiri. Saat Anda mengintegrasikan Amazon Q ke dalam siklus hidup pengembangan, lihat panduan ini untuk praktik terbaik, pemecahan masalah, dan contoh dunia nyata.

[Selain itu, tetap terinformasi dengan meninjau AWS blog dan panduan pengembang yang direferensikan di Sumber Daya.](#) Sumber daya ini menyediakan pembaruan, praktik terbaik, dan wawasan terbaru untuk membantu Anda mengoptimalkan penggunaan Pengembang Amazon Q.

Umpan balik Anda sangat berharga untuk meningkatkan panduan ini dan membantunya tetap menjadi sumber daya yang berharga bagi pengembang. Bagikan pengalaman, tantangan, dan saran Anda untuk versi masa depan. Masukan Anda akan membantu menyempurnakan panduan dengan contoh tambahan, skenario pemecahan masalah, dan wawasan yang disesuaikan dengan kebutuhan Anda.

# Sumber daya

## AWS blog

- [Mengakselerasi Siklus Hidup Pengembangan Perangkat Lunak Anda dengan Amazon Q](#)
- [Menata ulang pengembangan perangkat lunak dengan Agen Pengembang Amazon Q](#)
- [Lima contoh pemecahan masalah dengan Amazon Q](#)
- [Sesuaikan Pengembang Amazon Q di Anda IDE dengan basis kode pribadi Anda](#)
- [Tiga cara agen Pengembang Amazon Q untuk transformasi kode mempercepat peningkatan Java](#)
- [Memanfaatkan Pengembang Amazon Q untuk Debugging dan Pemeliharaan Kode yang Efisien](#)
- [Menguji aplikasi Anda dengan Amazon Q Developer](#)

## AWS dokumentasi

- [Panduan Pengguna Pengembang Amazon Q](#)
- [Kustomisasi kode Pengembang Amazon Q](#)
- [Transformasi kode Pengembang Amazon Q](#)

## AWS lokakarya

- [Hari Perendaman Pengembang Amazon Q](#)
- [Lokakarya Pengembang Amazon Q - Membangun Aplikasi Q-Words](#)
- [Lokakarya Pengembang Amazon Q - Membuat Prompt Efektif](#)

# Kontributor

Individu-individu berikut berkontribusi pada panduan ini:

- Joe King, Ilmuwan Data Senior, AWS
- Prateek Gupta, Ketua Tim — Sr. CAA, AWS
- Manohar Reddy Arranagu, Arsitek, DevOps AWS
- Soumik Roy, Arsitek Aplikasi Cloud, AWS
- Sanket Shinde, Konsultan, AWS

## Riwayat dokumen

Tabel berikut menjelaskan perubahan signifikan pada panduan ini. Jika Anda ingin diberi tahu tentang pembaruan masa depan, Anda dapat berlangganan [RSSumpnan](#).

Perubahan	Deskripsi	Tanggal
<a href="#">Publikasi awal</a>	—	16 Agustus 2024

# AWS Glosarium Panduan Preskriptif

Berikut ini adalah istilah yang umum digunakan dalam strategi, panduan, dan pola yang disediakan oleh Panduan AWS Preskriptif. Untuk menyarankan entri, silakan gunakan tautan Berikan umpan balik di akhir glosarium.

## Nomor

### 7 Rs

Tujuh strategi migrasi umum untuk memindahkan aplikasi ke cloud. Strategi ini dibangun di atas 5 Rs yang diidentifikasi Gartner pada tahun 2011 dan terdiri dari yang berikut:

- Refactor/Re-Architect — Memindahkan aplikasi dan memodifikasi arsitekturnya dengan memanfaatkan sepenuhnya fitur cloud-native untuk meningkatkan kelincahan, kinerja, dan skalabilitas. Ini biasanya melibatkan porting sistem operasi dan database. Contoh: Migrasikan database Oracle lokal Anda ke Amazon Aurora PostgreSQL Compatible Edition.
- Replatform (angkat dan bentuk ulang) — Pindahkan aplikasi ke cloud, dan perkenalkan beberapa tingkat pengoptimalan untuk memanfaatkan kemampuan cloud. Contoh: Migrasikan database Oracle lokal Anda ke Amazon Relational Database Service (Amazon RDS) untuk Oracle di AWS Cloud
- Pembelian kembali (drop and shop) - Beralih ke produk yang berbeda, biasanya dengan beralih dari lisensi tradisional ke model SaaS. Contoh: Migrasikan sistem manajemen hubungan pelanggan (CRM) Anda ke Salesforce.com.
- Rehost (lift dan shift) — Pindahkan aplikasi ke cloud tanpa membuat perubahan apa pun untuk memanfaatkan kemampuan cloud. Contoh: Migrasikan database Oracle lokal Anda ke Oracle pada instans EC2 di AWS Cloud
- Relokasi (hypervisor-level lift and shift) — Pindahkan infrastruktur ke cloud tanpa membeli perangkat keras baru, menulis ulang aplikasi, atau memodifikasi operasi yang ada. Anda memigrasikan server dari platform lokal ke layanan cloud untuk platform yang sama. Contoh: Migrasikan Microsoft Hyper-V aplikasi ke AWS.
- Pertahankan (kunjungi kembali) - Simpan aplikasi di lingkungan sumber Anda. Ini mungkin termasuk aplikasi yang memerlukan refactoring besar, dan Anda ingin menunda pekerjaan itu sampai nanti, dan aplikasi lama yang ingin Anda pertahankan, karena tidak ada pembenaran bisnis untuk memigrasikannya.

- Pensiun — Menonaktifkan atau menghapus aplikasi yang tidak lagi diperlukan di lingkungan sumber Anda.

## A

### ABAC

Lihat [kontrol akses berbasis atribut](#).

### layanan abstrak

Lihat [layanan terkelola](#).

### ASAM

Lihat [atomisitas, konsistensi, isolasi, daya tahan](#).

### migrasi aktif-aktif

Metode migrasi database di mana database sumber dan target tetap sinkron (dengan menggunakan alat replikasi dua arah atau operasi penulisan ganda), dan kedua database menangani transaksi dari menghubungkan aplikasi selama migrasi. Metode ini mendukung migrasi dalam batch kecil yang terkontrol alih-alih memerlukan pemotongan satu kali. Ini lebih fleksibel tetapi membutuhkan lebih banyak pekerjaan daripada migrasi [aktif-pasif](#).

### migrasi aktif-pasif

Metode migrasi database di mana database sumber dan target disimpan dalam sinkron, tetapi hanya database sumber yang menangani transaksi dari menghubungkan aplikasi sementara data direplikasi ke database target. Basis data target tidak menerima transaksi apa pun selama migrasi.

### fungsi agregat

Fungsi SQL yang beroperasi pada sekelompok baris dan menghitung nilai pengembalian tunggal untuk grup. Contoh fungsi agregat meliputi SUM dan MAX.

## AI

Lihat [kecerdasan buatan](#).

### AIOps

Lihat [operasi kecerdasan buatan](#).

## anonimisasi

Proses menghapus informasi pribadi secara permanen dalam kumpulan data. Anonimisasi dapat membantu melindungi privasi pribadi. Data anonim tidak lagi dianggap sebagai data pribadi.

## anti-pola

Solusi yang sering digunakan untuk masalah berulang di mana solusinya kontra-produktif, tidak efektif, atau kurang efektif daripada alternatif.

## kontrol aplikasi

Pendekatan keamanan yang memungkinkan penggunaan hanya aplikasi yang disetujui untuk membantu melindungi sistem dari malware.

## portofolio aplikasi

Kumpulan informasi rinci tentang setiap aplikasi yang digunakan oleh organisasi, termasuk biaya untuk membangun dan memelihara aplikasi, dan nilai bisnisnya. Informasi ini adalah kunci untuk [penemuan portofolio dan proses analisis dan](#) membantu mengidentifikasi dan memprioritaskan aplikasi yang akan dimigrasi, dimodernisasi, dan dioptimalkan.

## kecerdasan buatan (AI)

Bidang ilmu komputer yang didedikasikan untuk menggunakan teknologi komputasi untuk melakukan fungsi kognitif yang biasanya terkait dengan manusia, seperti belajar, memecahkan masalah, dan mengenali pola. Untuk informasi lebih lanjut, lihat [Apa itu Kecerdasan Buatan?](#)

## operasi kecerdasan buatan (AIOps)

Proses menggunakan teknik pembelajaran mesin untuk memecahkan masalah operasional, mengurangi insiden operasional dan intervensi manusia, dan meningkatkan kualitas layanan. Untuk informasi selengkapnya tentang cara AIOps digunakan dalam strategi AWS migrasi, lihat [panduan integrasi operasi](#).

## enkripsi asimetris

Algoritma enkripsi yang menggunakan sepasang kunci, kunci publik untuk enkripsi dan kunci pribadi untuk dekripsi. Anda dapat berbagi kunci publik karena tidak digunakan untuk dekripsi, tetapi akses ke kunci pribadi harus sangat dibatasi.

## atomisitas, konsistensi, isolasi, daya tahan (ACID)

Satu set properti perangkat lunak yang menjamin validitas data dan keandalan operasional database, bahkan dalam kasus kesalahan, kegagalan daya, atau masalah lainnya.

## kontrol akses berbasis atribut (ABAC)

Praktik membuat izin berbutir halus berdasarkan atribut pengguna, seperti departemen, peran pekerjaan, dan nama tim. Untuk informasi selengkapnya, lihat [ABAC untuk AWS](#) dokumentasi AWS Identity and Access Management (IAM).

## sumber data otoritatif

Lokasi di mana Anda menyimpan versi utama data, yang dianggap sebagai sumber informasi yang paling dapat diandalkan. Anda dapat menyalin data dari sumber data otoritatif ke lokasi lain untuk tujuan memproses atau memodifikasi data, seperti menganonimkan, menyunting, atau membuat nama samaran.

## Zona Ketersediaan

Lokasi berbeda di dalam AWS Region yang terisolasi dari kegagalan di Availability Zone lainnya dan menyediakan konektivitas jaringan latensi rendah yang murah ke Availability Zone lainnya di Wilayah yang sama.

## AWS Kerangka Adopsi Cloud (AWS CAF)

Kerangka pedoman dan praktik terbaik AWS untuk membantu organisasi mengembangkan rencana yang efisien dan efektif untuk bergerak dengan sukses ke cloud. AWS CAF mengatur panduan ke dalam enam area fokus yang disebut perspektif: bisnis, orang, tata kelola, platform, keamanan, dan operasi. Perspektif bisnis, orang, dan tata kelola fokus pada keterampilan dan proses bisnis; perspektif platform, keamanan, dan operasi fokus pada keterampilan dan proses teknis. Misalnya, perspektif masyarakat menargetkan pemangku kepentingan yang menangani sumber daya manusia (SDM), fungsi kepegawaian, dan manajemen orang. Untuk perspektif ini, AWS CAF memberikan panduan untuk pengembangan, pelatihan, dan komunikasi orang untuk membantu mempersiapkan organisasi untuk adopsi cloud yang sukses. Untuk informasi lebih lanjut, lihat [situs web AWS CAF dan whitepaper AWS CAF](#).

## AWS Kerangka Kualifikasi Beban Kerja (AWS WQF)

Alat yang mengevaluasi beban kerja migrasi database, merekomendasikan strategi migrasi, dan memberikan perkiraan kerja. AWS WQF disertakan dengan AWS Schema Conversion Tool (AWS SCT). Ini menganalisis skema database dan objek kode, kode aplikasi, dependensi, dan karakteristik kinerja, dan memberikan laporan penilaian.

## B

### bot buruk

[Bot](#) yang dimaksudkan untuk mengganggu atau menyebabkan kerugian bagi individu atau organisasi.

### BCP

Lihat [perencanaan kontinuitas bisnis](#).

### grafik perilaku

Pandangan interaktif yang terpadu tentang perilaku dan interaksi sumber daya dari waktu ke waktu. Anda dapat menggunakan grafik perilaku dengan Amazon Detective untuk memeriksa upaya logon yang gagal, panggilan API yang mencurigakan, dan tindakan serupa. Untuk informasi selengkapnya, lihat [Data dalam grafik perilaku](#) di dokumentasi Detektif.

### sistem big-endian

Sistem yang menyimpan byte paling signifikan terlebih dahulu. Lihat juga [endianness](#).

### klasifikasi biner

Sebuah proses yang memprediksi hasil biner (salah satu dari dua kelas yang mungkin). Misalnya, model ML Anda mungkin perlu memprediksi masalah seperti “Apakah email ini spam atau bukan spam?” atau “Apakah produk ini buku atau mobil?”

### filter mekar

Struktur data probabilistik dan efisien memori yang digunakan untuk menguji apakah suatu elemen adalah anggota dari suatu himpunan.

### deployment biru/hijau

Strategi penyebaran tempat Anda membuat dua lingkungan yang terpisah namun identik. Anda menjalankan versi aplikasi saat ini di satu lingkungan (biru) dan versi aplikasi baru di lingkungan lain (hijau). Strategi ini membantu Anda dengan cepat memutar kembali dengan dampak minimal.

### bot

Aplikasi perangkat lunak yang menjalankan tugas otomatis melalui internet dan mensimulasikan aktivitas atau interaksi manusia. Beberapa bot berguna atau bermanfaat, seperti perayap web yang mengindeks informasi di internet. Beberapa bot lain, yang dikenal sebagai bot buruk, dimaksudkan untuk mengganggu atau membahayakan individu atau organisasi.

## botnet

Jaringan [bot](#) yang terinfeksi oleh [malware](#) dan berada di bawah kendali satu pihak, yang dikenal sebagai bot herder atau operator bot. Botnet adalah mekanisme paling terkenal untuk skala bot dan dampaknya.

## cabang

Area berisi repositori kode. Cabang pertama yang dibuat dalam repositori adalah cabang utama. Anda dapat membuat cabang baru dari cabang yang ada, dan Anda kemudian dapat mengembangkan fitur atau memperbaiki bug di cabang baru. Cabang yang Anda buat untuk membangun fitur biasanya disebut sebagai cabang fitur. Saat fitur siap dirilis, Anda menggabungkan cabang fitur kembali ke cabang utama. Untuk informasi selengkapnya, lihat [Tentang cabang](#) (GitHub dokumentasi).

## akses break-glass

Dalam keadaan luar biasa dan melalui proses yang disetujui, cara cepat bagi pengguna untuk mendapatkan akses ke Akun AWS yang biasanya tidak memiliki izin untuk mengaksesnya. Untuk informasi lebih lanjut, lihat indikator [Implementasikan prosedur break-glass](#) dalam panduan Well-Architected AWS .

## strategi brownfield

Infrastruktur yang ada di lingkungan Anda. Saat mengadopsi strategi brownfield untuk arsitektur sistem, Anda merancang arsitektur di sekitar kendala sistem dan infrastruktur saat ini. Jika Anda memperluas infrastruktur yang ada, Anda dapat memadukan strategi brownfield dan [greenfield](#).

## cache penyangga

Area memori tempat data yang paling sering diakses disimpan.

## kemampuan bisnis

Apa yang dilakukan bisnis untuk menghasilkan nilai (misalnya, penjualan, layanan pelanggan, atau pemasaran). Arsitektur layanan mikro dan keputusan pengembangan dapat didorong oleh kemampuan bisnis. Untuk informasi selengkapnya, lihat bagian [Terorganisir di sekitar kemampuan bisnis](#) dari [Menjalankan layanan mikro kontainer](#) di whitepaper. AWS

## perencanaan kelangsungan bisnis (BCP)

Rencana yang membahas dampak potensial dari peristiwa yang mengganggu, seperti migrasi skala besar, pada operasi dan memungkinkan bisnis untuk melanjutkan operasi dengan cepat.

# C

## KAFE

Lihat [Kerangka Adopsi AWS Cloud](#).

### penyebaran kenari

Rilis versi yang lambat dan bertahap untuk pengguna akhir. Ketika Anda yakin, Anda menyebarkan versi baru dan mengganti versi saat ini secara keseluruhan.

## CCoE

Lihat [Cloud Center of Excellence](#).

## CDC

Lihat [mengubah pengambilan data](#).

### ubah pengambilan data (CDC)

Proses melacak perubahan ke sumber data, seperti tabel database, dan merekam metadata tentang perubahan tersebut. Anda dapat menggunakan CDC untuk berbagai tujuan, seperti mengaudit atau mereplikasi perubahan dalam sistem target untuk mempertahankan sinkronisasi.

### rekayasa kekacauan

Sengaja memperkenalkan kegagalan atau peristiwa yang mengganggu untuk menguji ketahanan sistem. Anda dapat menggunakan [AWS Fault Injection Service \(AWS FIS\)](#) untuk melakukan eksperimen yang menekankan AWS beban kerja Anda dan mengevaluasi responsnya.

## CI/CD

Lihat [integrasi berkelanjutan dan pengiriman berkelanjutan](#).

### klasifikasi

Proses kategorisasi yang membantu menghasilkan prediksi. Model ML untuk masalah klasifikasi memprediksi nilai diskrit. Nilai diskrit selalu berbeda satu sama lain. Misalnya, model mungkin perlu mengevaluasi apakah ada mobil dalam gambar atau tidak.

### Enkripsi sisi klien

Enkripsi data secara lokal, sebelum target Layanan AWS menerimanya.

## Pusat Keunggulan Cloud (CCoE)

Tim multi-disiplin yang mendorong upaya adopsi cloud di seluruh organisasi, termasuk mengembangkan praktik terbaik cloud, memobilisasi sumber daya, menetapkan jadwal migrasi, dan memimpin organisasi melalui transformasi skala besar. Untuk informasi selengkapnya, lihat [posting CCoE](#) di Blog Strategi AWS Cloud Perusahaan.

### komputasi cloud

Teknologi cloud yang biasanya digunakan untuk penyimpanan data jarak jauh dan manajemen perangkat IoT. Cloud computing umumnya terhubung ke teknologi [edge computing](#).

### model operasi cloud

Dalam organisasi TI, model operasi yang digunakan untuk membangun, mematangkan, dan mengoptimalkan satu atau lebih lingkungan cloud. Untuk informasi selengkapnya, lihat [Membangun Model Operasi Cloud Anda](#).

### tahap adopsi cloud

Empat fase yang biasanya dilalui organisasi ketika mereka bermigrasi ke AWS Cloud:

- Proyek — Menjalankan beberapa proyek terkait cloud untuk bukti konsep dan tujuan pembelajaran
- Foundation — Melakukan investasi dasar untuk meningkatkan adopsi cloud Anda (misalnya, membuat landing zone, mendefinisikan CCoE, membuat model operasi)
- Migrasi — Migrasi aplikasi individual
- Re-invention — Mengoptimalkan produk dan layanan, dan berinovasi di cloud

Tahapan ini didefinisikan oleh Stephen Orban dalam posting blog [The Journey Toward Cloud-First & the Stages of Adoption](#) di blog Strategi Perusahaan. AWS Cloud Untuk informasi tentang bagaimana kaitannya dengan strategi AWS migrasi, lihat [panduan kesiapan migrasi](#).

### CMDB

Lihat [database manajemen konfigurasi](#).

### repositori kode

Lokasi di mana kode sumber dan aset lainnya, seperti dokumentasi, sampel, dan skrip, disimpan dan diperbarui melalui proses kontrol versi. Repositori cloud umum termasuk GitHub atau Bitbucket Cloud. Setiap versi kode disebut cabang. Dalam struktur layanan mikro, setiap repositori

dikhususkan untuk satu bagian fungsionalitas. Pipa CI/CD tunggal dapat menggunakan beberapa repositori.

#### cache dingin

Cache buffer yang kosong, tidak terisi dengan baik, atau berisi data basi atau tidak relevan. Ini mempengaruhi kinerja karena instance database harus membaca dari memori utama atau disk, yang lebih lambat daripada membaca dari cache buffer.

#### data dingin

Data yang jarang diakses dan biasanya historis. Saat menanyakan jenis data ini, kueri lambat biasanya dapat diterima. Memindahkan data ini ke tingkat penyimpanan atau kelas yang berkinerja lebih rendah dan lebih murah dapat mengurangi biaya.

#### visi komputer (CV)

Bidang [AI](#) yang menggunakan pembelajaran mesin untuk menganalisis dan mengekstrak informasi dari format visual seperti gambar dan video digital. Misalnya, Amazon SageMaker AI menyediakan algoritma pemrosesan gambar untuk CV.

#### konfigurasi drift

Untuk beban kerja, konfigurasi berubah dari status yang diharapkan. Ini dapat menyebabkan beban kerja menjadi tidak patuh, dan biasanya bertahap dan tidak disengaja.

#### database manajemen konfigurasi (CMDB)

Repositori yang menyimpan dan mengelola informasi tentang database dan lingkungan TI, termasuk komponen perangkat keras dan perangkat lunak dan konfigurasinya. Anda biasanya menggunakan data dari CMDB dalam penemuan portofolio dan tahap analisis migrasi.

#### paket kesesuaian

Kumpulan AWS Config aturan dan tindakan remediasi yang dapat Anda kumpulkan untuk menyesuaikan kepatuhan dan pemeriksaan keamanan Anda. Anda dapat menerapkan paket kesesuaian sebagai entitas tunggal di Akun AWS dan Region, atau di seluruh organisasi, dengan menggunakan templat YAMM. Untuk informasi selengkapnya, lihat [Paket kesesuaian dalam dokumentasi](#). AWS Config

#### integrasi berkelanjutan dan pengiriman berkelanjutan (CI/CD)

Proses mengotomatiskan sumber, membangun, menguji, pementasan, dan tahap produksi dari proses rilis perangkat lunak. CI/CD biasanya digambarkan sebagai pipa. CI/CD dapat membantu

Anda mengotomatiskan proses, meningkatkan produktivitas, meningkatkan kualitas kode, dan memberikan lebih cepat. Untuk informasi lebih lanjut, lihat [Manfaat pengiriman berkelanjutan](#). CD juga dapat berarti penerapan berkelanjutan. Untuk informasi selengkapnya, lihat [Continuous Delivery vs Continuous Deployment](#).

## CV

Lihat [visi komputer](#).

## D

### data saat istirahat

Data yang stasioner di jaringan Anda, seperti data yang ada di penyimpanan.

### klasifikasi data

Proses untuk mengidentifikasi dan mengkategorikan data dalam jaringan Anda berdasarkan kekritisannya dan sensitivitasnya. Ini adalah komponen penting dari setiap strategi manajemen risiko keamanan siber karena membantu Anda menentukan perlindungan dan kontrol retensi yang tepat untuk data. Klasifikasi data adalah komponen pilar keamanan dalam AWS Well-Architected Framework. Untuk informasi selengkapnya, lihat [Klasifikasi data](#).

### penyimpangan data

Variasi yang berarti antara data produksi dan data yang digunakan untuk melatih model ML, atau perubahan yang berarti dalam data input dari waktu ke waktu. Penyimpangan data dapat mengurangi kualitas, akurasi, dan keadilan keseluruhan dalam prediksi model ML.

### data dalam transit

Data yang aktif bergerak melalui jaringan Anda, seperti antara sumber daya jaringan.

### jala data

Kerangka arsitektur yang menyediakan kepemilikan data terdistribusi dan terdesentralisasi dengan manajemen dan tata kelola terpusat.

### minimalisasi data

Prinsip pengumpulan dan pemrosesan hanya data yang sangat diperlukan. Mempraktikkan minimalisasi data di dalamnya AWS Cloud dapat mengurangi risiko privasi, biaya, dan jejak karbon analitik Anda.

## perimeter data

Satu set pagar pembatas pencegahan di AWS lingkungan Anda yang membantu memastikan bahwa hanya identitas tepercaya yang mengakses sumber daya tepercaya dari jaringan yang diharapkan. Untuk informasi selengkapnya, lihat [Membangun perimeter data pada AWS](#).

## prapemrosesan data

Untuk mengubah data mentah menjadi format yang mudah diuraikan oleh model ML Anda. Preprocessing data dapat berarti menghapus kolom atau baris tertentu dan menangani nilai yang hilang, tidak konsisten, atau duplikat.

## asal data

Proses melacak asal dan riwayat data sepanjang siklus hidupnya, seperti bagaimana data dihasilkan, ditransmisikan, dan disimpan.

## subjek data

Individu yang datanya dikumpulkan dan diproses.

## gudang data

Sistem manajemen data yang mendukung intelijen bisnis, seperti analitik. Gudang data biasanya berisi sejumlah besar data historis, dan biasanya digunakan untuk kueri dan analisis.

## bahasa definisi database (DDL)

Pernyataan atau perintah untuk membuat atau memodifikasi struktur tabel dan objek dalam database.

## bahasa manipulasi basis data (DHTML)

Pernyataan atau perintah untuk memodifikasi (memasukkan, memperbarui, dan menghapus) informasi dalam database.

## DDL

Lihat [bahasa definisi database](#).

## ansambel yang dalam

Untuk menggabungkan beberapa model pembelajaran mendalam untuk prediksi. Anda dapat menggunakan ansambel dalam untuk mendapatkan prediksi yang lebih akurat atau untuk memperkirakan ketidakpastian dalam prediksi.

## pembelajaran mendalam

Subbidang ML yang menggunakan beberapa lapisan jaringan saraf tiruan untuk mengidentifikasi pemetaan antara data input dan variabel target yang diinginkan.

## defense-in-depth

Pendekatan keamanan informasi di mana serangkaian mekanisme dan kontrol keamanan dilapisi dengan cermat di seluruh jaringan komputer untuk melindungi kerahasiaan, integritas, dan ketersediaan jaringan dan data di dalamnya. Saat Anda mengadopsi strategi ini AWS, Anda menambahkan beberapa kontrol pada lapisan AWS Organizations struktur yang berbeda untuk membantu mengamankan sumber daya. Misalnya, defense-in-depth pendekatan mungkin menggabungkan otentikasi multi-faktor, segmentasi jaringan, dan enkripsi.

## administrator yang didelegasikan

Di AWS Organizations, layanan yang kompatibel dapat mendaftarkan akun AWS anggota untuk mengelola akun organisasi dan mengelola izin untuk layanan tersebut. Akun ini disebut administrator yang didelegasikan untuk layanan itu. Untuk informasi selengkapnya dan daftar layanan yang kompatibel, lihat [Layanan yang berfungsi dengan AWS Organizations](#) AWS Organizations dokumentasi.

## deployment

Proses pembuatan aplikasi, fitur baru, atau perbaikan kode tersedia di lingkungan target. Deployment melibatkan penerapan perubahan dalam basis kode dan kemudian membangun dan menjalankan basis kode itu di lingkungan aplikasi.

## lingkungan pengembangan

Lihat [lingkungan](#).

## kontrol detektif

Kontrol keamanan yang dirancang untuk mendeteksi, mencatat, dan memperingatkan setelah suatu peristiwa terjadi. Kontrol ini adalah garis pertahanan kedua, memperingatkan Anda tentang peristiwa keamanan yang melewati kontrol pencegahan yang ada. Untuk informasi selengkapnya, lihat Kontrol [Detektif dalam Menerapkan kontrol](#) keamanan pada. AWS

## pemetaan aliran nilai pengembangan (DVSM)

Sebuah proses yang digunakan untuk mengidentifikasi dan memprioritaskan kendala yang mempengaruhi kecepatan dan kualitas dalam siklus hidup pengembangan perangkat lunak. DVSM memperluas proses pemetaan aliran nilai yang awalnya dirancang untuk praktik

manufaktur ramping. Ini berfokus pada langkah-langkah dan tim yang diperlukan untuk menciptakan dan memindahkan nilai melalui proses pengembangan perangkat lunak.

## kembar digital

Representasi virtual dari sistem dunia nyata, seperti bangunan, pabrik, peralatan industri, atau jalur produksi. Kembar digital mendukung pemeliharaan prediktif, pemantauan jarak jauh, dan optimalisasi produksi.

## tabel dimensi

Dalam [skema bintang](#), tabel yang lebih kecil yang berisi atribut data tentang data kuantitatif dalam tabel fakta. Atribut tabel dimensi biasanya bidang teks atau angka diskrit yang berperilaku seperti teks. Atribut ini biasanya digunakan untuk pembatasan kueri, pemfilteran, dan pelabelan set hasil.

## musibah

Peristiwa yang mencegah beban kerja atau sistem memenuhi tujuan bisnisnya di lokasi utama yang digunakan. Peristiwa ini dapat berupa bencana alam, kegagalan teknis, atau akibat dari tindakan manusia, seperti kesalahan konfigurasi yang tidak disengaja atau serangan malware.

## pemulihan bencana (DR)

Strategi dan proses yang Anda gunakan untuk meminimalkan downtime dan kehilangan data yang disebabkan oleh [bencana](#). Untuk informasi selengkapnya, lihat [Disaster Recovery of Workloads on AWS: Recovery in the Cloud in the AWS Well-Architected Framework](#).

## DML~

Lihat [bahasa manipulasi basis data](#).

## desain berbasis domain

Pendekatan untuk mengembangkan sistem perangkat lunak yang kompleks dengan menghubungkan komponennya ke domain yang berkembang, atau tujuan bisnis inti, yang dilayani oleh setiap komponen. Konsep ini diperkenalkan oleh Eric Evans dalam bukunya, *Domain-Driven Design: Tackling Complexity in the Heart of Software* (Boston: Addison-Wesley Professional, 2003). Untuk informasi tentang cara menggunakan desain berbasis domain dengan pola gambar pencekik, lihat Memodernisasi layanan web [Microsoft ASP.NET \(ASMX\) lama secara bertahap menggunakan container dan Amazon API Gateway](#).

## DR

Lihat [pemulihan bencana](#).

## deteksi drift

Melacak penyimpangan dari konfigurasi dasar. Misalnya, Anda dapat menggunakan AWS CloudFormation untuk [mendeteksi penyimpangan dalam sumber daya sistem](#), atau Anda dapat menggunakannya AWS Control Tower untuk [mendeteksi perubahan di landing zone](#) yang mungkin memengaruhi kepatuhan terhadap persyaratan tata kelola.

## DVSM

Lihat [pemetaan aliran nilai pengembangan](#).

## E

### EDA

Lihat [analisis data eksplorasi](#).

### EDI

Lihat [pertukaran data elektronik](#).

## komputasi tepi

Teknologi yang meningkatkan daya komputasi untuk perangkat pintar di tepi jaringan IoT. Jika dibandingkan dengan [komputasi awan](#), komputasi tepi dapat mengurangi latensi komunikasi dan meningkatkan waktu respons.

## pertukaran data elektronik (EDI)

Pertukaran otomatis dokumen bisnis antar organisasi. Untuk informasi selengkapnya, lihat [Apa itu Pertukaran Data Elektronik](#).

## enkripsi

Proses komputasi yang mengubah data plaintext, yang dapat dibaca manusia, menjadi ciphertext.

## kunci enkripsi

String kriptografi dari bit acak yang dihasilkan oleh algoritma enkripsi. Panjang kunci dapat bervariasi, dan setiap kunci dirancang agar tidak dapat diprediksi dan unik.

## endianness

Urutan byte disimpan dalam memori komputer. Sistem big-endian menyimpan byte paling signifikan terlebih dahulu. Sistem little-endian menyimpan byte paling tidak signifikan terlebih dahulu.

## titik akhir

Lihat [titik akhir layanan](#).

## layanan endpoint

Layanan yang dapat Anda host di cloud pribadi virtual (VPC) untuk dibagikan dengan pengguna lain. Anda dapat membuat layanan endpoint dengan AWS PrivateLink dan memberikan izin kepada prinsipal lain Akun AWS atau ke AWS Identity and Access Management (IAM). Akun atau prinsipal ini dapat terhubung ke layanan endpoint Anda secara pribadi dengan membuat titik akhir VPC antarmuka. Untuk informasi selengkapnya, lihat [Membuat layanan titik akhir](#) di dokumentasi Amazon Virtual Private Cloud (Amazon VPC).

## perencanaan sumber daya perusahaan (ERP)

Sistem yang mengotomatiskan dan mengelola proses bisnis utama (seperti akuntansi, [MES](#), dan manajemen proyek) untuk suatu perusahaan.

## enkripsi amplop

Proses mengenkripsi kunci enkripsi dengan kunci enkripsi lain. Untuk informasi selengkapnya, lihat [Enkripsi amplop](#) dalam dokumentasi AWS Key Management Service (AWS KMS).

## lingkungan

Sebuah contoh dari aplikasi yang sedang berjalan. Berikut ini adalah jenis lingkungan yang umum dalam komputasi awan:

- **Development Environment** — Sebuah contoh dari aplikasi yang berjalan yang hanya tersedia untuk tim inti yang bertanggung jawab untuk memelihara aplikasi. Lingkungan pengembangan digunakan untuk menguji perubahan sebelum mempromosikannya ke lingkungan atas. Jenis lingkungan ini kadang-kadang disebut sebagai lingkungan pengujian.
- **lingkungan yang lebih rendah** — Semua lingkungan pengembangan untuk aplikasi, seperti yang digunakan untuk build awal dan pengujian.
- **lingkungan produksi** — Sebuah contoh dari aplikasi yang berjalan yang dapat diakses oleh pengguna akhir. Dalam sebuah CI/CD pipeline, lingkungan produksi adalah lingkungan penyebaran terakhir.

- lingkungan atas — Semua lingkungan yang dapat diakses oleh pengguna selain tim pengembangan inti. Ini dapat mencakup lingkungan produksi, lingkungan praproduksi, dan lingkungan untuk pengujian penerimaan pengguna.

## epik

Dalam metodologi tangkas, kategori fungsional yang membantu mengatur dan memprioritaskan pekerjaan Anda. Epik memberikan deskripsi tingkat tinggi tentang persyaratan dan tugas implementasi. Misalnya, epos keamanan AWS CAF mencakup manajemen identitas dan akses, kontrol detektif, keamanan infrastruktur, perlindungan data, dan respons insiden. Untuk informasi selengkapnya tentang epos dalam strategi AWS migrasi, lihat [panduan implementasi program](#).

## ERP

Lihat [perencanaan sumber daya perusahaan](#).

## analisis data eksplorasi (EDA)

Proses menganalisis dataset untuk memahami karakteristik utamanya. Anda mengumpulkan atau mengumpulkan data dan kemudian melakukan penyelidikan awal untuk menemukan pola, mendeteksi anomali, dan memeriksa asumsi. EDA dilakukan dengan menghitung statistik ringkasan dan membuat visualisasi data.

## F

### tabel fakta

Tabel tengah dalam [skema bintang](#). Ini menyimpan data kuantitatif tentang operasi bisnis. Biasanya, tabel fakta berisi dua jenis kolom: kolom yang berisi ukuran dan yang berisi kunci asing ke tabel dimensi.

### gagal cepat

Filosofi yang menggunakan pengujian yang sering dan bertahap untuk mengurangi siklus hidup pengembangan. Ini adalah bagian penting dari pendekatan tangkas.

### batas isolasi kesalahan

Dalam AWS Cloud, batas seperti Availability Zone, AWS Region, control plane, atau data plane yang membatasi efek kegagalan dan membantu meningkatkan ketahanan beban kerja. Untuk informasi selengkapnya, lihat [Batas Isolasi AWS Kesalahan](#).

## cabang fitur

Lihat [cabang](#).

## fitur

Data input yang Anda gunakan untuk membuat prediksi. Misalnya, dalam konteks manufaktur, fitur bisa berupa gambar yang diambil secara berkala dari lini manufaktur.

## pentingnya fitur

Seberapa signifikan fitur untuk prediksi model. Ini biasanya dinyatakan sebagai skor numerik yang dapat dihitung melalui berbagai teknik, seperti Shapley Additive Explanations (SHAP) dan gradien terintegrasi. Untuk informasi lebih lanjut, lihat [Interpretabilitas model pembelajaran mesin](#) dengan AWS

## transformasi fitur

Untuk mengoptimalkan data untuk proses ML, termasuk memperkaya data dengan sumber tambahan, menskalakan nilai, atau mengekstrak beberapa set informasi dari satu bidang data. Hal ini memungkinkan model ML untuk mendapatkan keuntungan dari data. Misalnya, jika Anda memecah tanggal "2021-05-27 00:15:37" menjadi "2021", "Mei", "Kamis", dan "15", Anda dapat membantu algoritme pembelajaran mempelajari pola bernuansa yang terkait dengan komponen data yang berbeda.

## beberapa tembakan mendorong

Menyediakan [LLM](#) dengan sejumlah kecil contoh yang menunjukkan tugas dan output yang diinginkan sebelum memintanya untuk melakukan tugas serupa. Teknik ini adalah aplikasi pembelajaran dalam konteks, di mana model belajar dari contoh (bidikan) yang tertanam dalam petunjuk. Beberapa bidikan dapat efektif untuk tugas-tugas yang memerlukan pemformatan, penalaran, atau pengetahuan domain tertentu. Lihat juga [bidikan nol](#).

## FGAC

Lihat kontrol [akses berbutir halus](#).

## kontrol akses berbutir halus (FGAC)

Penggunaan beberapa kondisi untuk mengizinkan atau menolak permintaan akses.

## migrasi flash-cut

Metode migrasi database yang menggunakan replikasi data berkelanjutan melalui [pengambilan data perubahan](#) untuk memigrasikan data dalam waktu sesingkat mungkin, alih-alih

menggunakan pendekatan bertahap. Tujuannya adalah untuk menjaga downtime seminimal mungkin.

## FM

Lihat [model pondasi](#).

### model pondasi (FM)

Jaringan saraf pembelajaran mendalam yang besar yang telah melatih kumpulan data besar-besaran data umum dan tidak berlabel. FMs mampu melakukan berbagai tugas umum, seperti memahami bahasa, menghasilkan teks dan gambar, dan berbicara dalam bahasa alami. Untuk informasi selengkapnya, lihat [Apa itu Model Foundation](#).

## G

### AI generatif

Subset model [AI](#) yang telah dilatih pada sejumlah besar data dan yang dapat menggunakan prompt teks sederhana untuk membuat konten dan artefak baru, seperti gambar, video, teks, dan audio. Untuk informasi lebih lanjut, lihat [Apa itu AI Generatif](#).

### pemblokiran geografis

Lihat [pembatasan geografis](#).

### pembatasan geografis (pemblokiran geografis)

Di Amazon CloudFront, opsi untuk mencegah pengguna di negara tertentu mengakses distribusi konten. Anda dapat menggunakan daftar izinkan atau daftar blokir untuk menentukan negara yang disetujui dan dilarang. Untuk informasi selengkapnya, lihat [Membatasi distribusi geografis konten Anda](#) dalam dokumentasi. CloudFront

### Alur kerja Gitflow

Pendekatan di mana lingkungan bawah dan atas menggunakan cabang yang berbeda dalam repositori kode sumber. Alur kerja Gitflow dianggap warisan, dan [alur kerja berbasis batang](#) adalah pendekatan modern yang lebih disukai.

### gambar emas

Sebuah snapshot dari sistem atau perangkat lunak yang digunakan sebagai template untuk menyebarkan instance baru dari sistem atau perangkat lunak itu. Misalnya, di bidang manufaktur,

gambar emas dapat digunakan untuk menyediakan perangkat lunak pada beberapa perangkat dan membantu meningkatkan kecepatan, skalabilitas, dan produktivitas dalam operasi manufaktur perangkat.

### strategi greenfield

Tidak adanya infrastruktur yang ada di lingkungan baru. [Saat mengadopsi strategi greenfield untuk arsitektur sistem, Anda dapat memilih semua teknologi baru tanpa batasan kompatibilitas dengan infrastruktur yang ada, juga dikenal sebagai brownfield.](#) Jika Anda memperluas infrastruktur yang ada, Anda dapat memadukan strategi brownfield dan greenfield.

### pagar pembatas

Aturan tingkat tinggi yang membantu mengatur sumber daya, kebijakan, dan kepatuhan di seluruh unit organisasi (OU). Pagar pembatas preventif menegakkan kebijakan untuk memastikan keselarasan dengan standar kepatuhan. Mereka diimplementasikan dengan menggunakan kebijakan kontrol layanan dan batas izin IAM. Detective guardrails mendeteksi pelanggaran kebijakan dan masalah kepatuhan, dan menghasilkan peringatan untuk remediasi. Mereka diimplementasikan dengan menggunakan AWS Config, AWS Security Hub CSPM, Amazon GuardDuty AWS Trusted Advisor, Amazon Inspector, dan pemeriksaan khusus AWS Lambda .

## H

### HA

Lihat [ketersediaan tinggi](#).

### migrasi database heterogen

Memigrasi database sumber Anda ke database target yang menggunakan mesin database yang berbeda (misalnya, Oracle ke Amazon Aurora). Migrasi heterogen biasanya merupakan bagian dari upaya arsitektur ulang, dan mengubah skema dapat menjadi tugas yang kompleks. [AWS menyediakan AWS SCT](#) yang membantu dengan konversi skema.

### ketersediaan tinggi (HA)

Kemampuan beban kerja untuk beroperasi terus menerus, tanpa intervensi, jika terjadi tantangan atau bencana. Sistem HA dirancang untuk gagal secara otomatis, secara konsisten memberikan kinerja berkualitas tinggi, dan menangani beban dan kegagalan yang berbeda dengan dampak kinerja minimal.

## modernisasi sejarawan

Pendekatan yang digunakan untuk memodernisasi dan meningkatkan sistem teknologi operasional (OT) untuk melayani kebutuhan industri manufaktur dengan lebih baik. Sejarawan adalah jenis database yang digunakan untuk mengumpulkan dan menyimpan data dari berbagai sumber di pabrik.

## data penahanan

Sebagian dari data historis berlabel yang ditahan dari kumpulan data yang digunakan untuk melatih model pembelajaran [mesin](#). Anda dapat menggunakan data penahanan untuk mengevaluasi kinerja model dengan membandingkan prediksi model dengan data penahanan.

## migrasi database homogen

Memigrasi database sumber Anda ke database target yang berbagi mesin database yang sama (misalnya, Microsoft SQL Server ke Amazon RDS for SQL Server). Migrasi homogen biasanya merupakan bagian dari upaya rehosting atau replatforming. Anda dapat menggunakan utilitas database asli untuk memigrasi skema.

## data panas

Data yang sering diakses, seperti data real-time atau data translasi terbaru. Data ini biasanya memerlukan tingkat atau kelas penyimpanan berkinerja tinggi untuk memberikan respons kueri yang cepat.

## perbaikan terbaru

Perbaikan mendesak untuk masalah kritis dalam lingkungan produksi. Karena urgensinya, perbaikan terbaru biasanya dibuat di luar alur kerja DevOps rilis biasa.

## periode hypercare

Segera setelah cutover, periode waktu ketika tim migrasi mengelola dan memantau aplikasi yang dimigrasi di cloud untuk mengatasi masalah apa pun. Biasanya, periode ini panjangnya 1-4 hari. Pada akhir periode hypercare, tim migrasi biasanya mentransfer tanggung jawab untuk aplikasi ke tim operasi cloud.

|

## IAC

Lihat [infrastruktur sebagai kode](#).

|

## kebijakan berbasis identitas

Kebijakan yang dilampirkan pada satu atau beberapa prinsip IAM yang mendefinisikan izin mereka dalam lingkungan. AWS Cloud

## aplikasi idle

Aplikasi yang memiliki penggunaan CPU dan memori rata-rata antara 5 dan 20 persen selama periode 90 hari. Dalam proyek migrasi, adalah umum untuk menghentikan aplikasi ini atau mempertahankannya di tempat.

## IIoT

Lihat [Internet of Things industri](#).

## infrastruktur yang tidak dapat diubah

Model yang menyebarkan infrastruktur baru untuk beban kerja produksi alih-alih memperbarui, menambal, atau memodifikasi infrastruktur yang ada. [Infrastruktur yang tidak dapat diubah secara inheren lebih konsisten, andal, dan dapat diprediksi daripada infrastruktur yang dapat berubah](#). Untuk informasi selengkapnya, lihat praktik terbaik [Deploy using immutable infrastructure](#) di AWS Well-Architected Framework.

## masuk (masuknya) VPC

Dalam arsitektur AWS multi-akun, VPC yang menerima, memeriksa, dan merutekan koneksi jaringan dari luar aplikasi. [Arsitektur Referensi AWS Keamanan](#) merekomendasikan pengaturan akun Jaringan Anda dengan inbound, outbound, dan inspeksi VPCs untuk melindungi antarmuka dua arah antara aplikasi Anda dan internet yang lebih luas.

## migrasi inkremental

Strategi cutover di mana Anda memigrasikan aplikasi Anda dalam bagian-bagian kecil alih-alih melakukan satu cutover penuh. Misalnya, Anda mungkin hanya memindahkan beberapa layanan mikro atau pengguna ke sistem baru pada awalnya. Setelah Anda memverifikasi bahwa semuanya berfungsi dengan baik, Anda dapat secara bertahap memindahkan layanan mikro atau pengguna tambahan hingga Anda dapat menonaktifkan sistem lama Anda. Strategi ini mengurangi risiko yang terkait dengan migrasi besar.

## Industri 4.0

Sebuah istilah yang diperkenalkan oleh [Klaus Schwab](#) pada tahun 2016 untuk merujuk pada modernisasi proses manufaktur melalui kemajuan dalam konektivitas, data real-time, otomatisasi, analitik, dan AI/ML.

## infrastruktur

Semua sumber daya dan aset yang terkandung dalam lingkungan aplikasi.

### infrastruktur sebagai kode (IAC)

Proses penyediaan dan pengelolaan infrastruktur aplikasi melalui satu set file konfigurasi. IAC dirancang untuk membantu Anda memusatkan manajemen infrastruktur, menstandarisasi sumber daya, dan menskalakan dengan cepat sehingga lingkungan baru dapat diulang, andal, dan konsisten.

### Internet of Things industri (IIoT)

Penggunaan sensor dan perangkat yang terhubung ke internet di sektor industri, seperti manufaktur, energi, otomotif, perawatan kesehatan, ilmu kehidupan, dan pertanian. Untuk informasi lebih lanjut, lihat [Membangun strategi transformasi digital Internet of Things \(IIoT\) industri](#).

### inspeksi VPC

Dalam arsitektur AWS multi-akun, VPC terpusat yang mengelola inspeksi lalu lintas jaringan antara VPCs (dalam yang sama atau berbeda Wilayah AWS), internet, dan jaringan lokal. [Arsitektur Referensi AWS Keamanan](#) merekomendasikan pengaturan akun Jaringan Anda dengan inbound, outbound, dan inspeksi VPCs untuk melindungi antarmuka dua arah antara aplikasi Anda dan internet yang lebih luas.

### Internet of Things (IoT)

Jaringan objek fisik yang terhubung dengan sensor atau prosesor tertanam yang berkomunikasi dengan perangkat dan sistem lain melalui internet atau melalui jaringan komunikasi lokal. Untuk informasi selengkapnya, lihat [Apa itu IoT?](#)

### interpretabilitas

Karakteristik model pembelajaran mesin yang menggambarkan sejauh mana manusia dapat memahami bagaimana prediksi model bergantung pada inputnya. Untuk informasi lebih lanjut, lihat [Interpretabilitas model pembelajaran mesin](#) dengan AWS

### IoT

Lihat [Internet of Things](#).

## Perpustakaan informasi TI (ITIL)

Serangkaian praktik terbaik untuk memberikan layanan TI dan menyelaraskan layanan ini dengan persyaratan bisnis. ITIL menyediakan dasar untuk ITSM.

## Manajemen layanan TI (ITSM)

Kegiatan yang terkait dengan merancang, menerapkan, mengelola, dan mendukung layanan TI untuk suatu organisasi. Untuk informasi tentang mengintegrasikan operasi cloud dengan alat ITSM, lihat panduan [integrasi operasi](#).

## ITIL

Lihat [perpustakaan informasi TI](#).

## ITSM

Lihat [manajemen layanan TI](#).

## L

### kontrol akses berbasis label (LBAC)

Implementasi kontrol akses wajib (MAC) di mana pengguna dan data itu sendiri masing-masing secara eksplisit diberi nilai label keamanan. Persimpangan antara label keamanan pengguna dan label keamanan data menentukan baris dan kolom mana yang dapat dilihat oleh pengguna.

### landing zone

Landing zone adalah AWS lingkungan multi-akun yang dirancang dengan baik yang dapat diskalakan dan aman. Ini adalah titik awal dari mana organisasi Anda dapat dengan cepat meluncurkan dan menyebarkan beban kerja dan aplikasi dengan percaya diri dalam lingkungan keamanan dan infrastruktur mereka. Untuk informasi selengkapnya tentang zona pendaratan, lihat [Menyiapkan lingkungan multi-akun AWS yang aman dan dapat diskalakan](#).

### model bahasa besar (LLM)

Model [AI](#) pembelajaran mendalam yang dilatih sebelumnya pada sejumlah besar data. LLM dapat melakukan beberapa tugas, seperti menjawab pertanyaan, meringkas dokumen, menerjemahkan teks ke dalam bahasa lain, dan menyelesaikan kalimat. Untuk informasi lebih lanjut, lihat [Apa itu LLMs](#).

## migrasi besar

Migrasi 300 atau lebih server.

## LBAC

Lihat [kontrol akses berbasis label](#).

## hak istimewa paling sedikit

Praktik keamanan terbaik untuk memberikan izin minimum yang diperlukan untuk melakukan tugas. Untuk informasi selengkapnya, lihat [Menerapkan izin hak istimewa terkecil dalam dokumentasi IAM](#).

## angkat dan geser

Lihat [7 Rs](#).

## sistem endian kecil

Sebuah sistem yang menyimpan byte paling tidak signifikan terlebih dahulu. Lihat juga [endianness](#).

## LLM

Lihat [model bahasa besar](#).

## lingkungan yang lebih rendah

Lihat [lingkungan](#).

# M

## pembelajaran mesin (ML)

Jenis kecerdasan buatan yang menggunakan algoritma dan teknik untuk pengenalan pola dan pembelajaran. ML menganalisis dan belajar dari data yang direkam, seperti data Internet of Things (IoT), untuk menghasilkan model statistik berdasarkan pola. Untuk informasi selengkapnya, lihat [Machine Learning](#).

## cabang utama

Lihat [cabang](#).

## malware

Perangkat lunak yang dirancang untuk membahayakan keamanan atau privasi komputer. Malware dapat mengganggu sistem komputer, membocorkan informasi sensitif, atau mendapatkan akses yang tidak sah. Contoh malware termasuk virus, worm, ransomware, Trojan horse, spyware, dan keyloggers.

## layanan terkelola

Layanan AWS yang AWS mengoperasikan lapisan infrastruktur, sistem operasi, dan platform, dan Anda mengakses titik akhir untuk menyimpan dan mengambil data. Amazon Simple Storage Service (Amazon S3) dan Amazon DynamoDB adalah contoh layanan terkelola. Ini juga dikenal sebagai layanan abstrak.

## sistem eksekusi manufaktur (MES)

Sistem perangkat lunak untuk melacak, memantau, mendokumentasikan, dan mengendalikan proses produksi yang mengubah bahan baku menjadi produk jadi di lantai toko.

## PETA

Lihat [Program Percepatan Migrasi](#).

## mekanisme

Proses lengkap di mana Anda membuat alat, mendorong adopsi alat, dan kemudian memeriksa hasilnya untuk melakukan penyesuaian. Mekanisme adalah siklus yang memperkuat dan meningkatkan dirinya sendiri saat beroperasi. Untuk informasi lebih lanjut, lihat [Membangun mekanisme](#) di AWS Well-Architected Framework.

## akun anggota

Semua Akun AWS selain akun manajemen yang merupakan bagian dari organisasi di AWS Organizations. Akun dapat menjadi anggota dari hanya satu organisasi pada suatu waktu.

## MES

Lihat [sistem eksekusi manufaktur](#).

## Transportasi Telemetri Antrian Pesan (MQTT)

[Protokol komunikasi ringan machine-to-machine \(M2M\), berdasarkan pola terbitkan/berlangganan, untuk perangkat IoT yang dibatasi sumber daya.](#)

## layanan mikro

Layanan kecil dan independen yang berkomunikasi dengan jelas APIs dan biasanya dimiliki oleh tim kecil yang mandiri. Misalnya, sistem asuransi mungkin mencakup layanan mikro yang memetakan kemampuan bisnis, seperti penjualan atau pemasaran, atau subdomain, seperti pembelian, klaim, atau analitik. Manfaat layanan mikro termasuk kelincahan, penskalaan yang fleksibel, penyebaran yang mudah, kode yang dapat digunakan kembali, dan ketahanan. Untuk informasi selengkapnya, lihat [Mengintegrasikan layanan mikro dengan menggunakan layanan tanpa AWS server](#).

## arsitektur microservices

Pendekatan untuk membangun aplikasi dengan komponen independen yang menjalankan setiap proses aplikasi sebagai layanan mikro. Layanan mikro ini berkomunikasi melalui antarmuka yang terdefinisi dengan baik dengan menggunakan ringan. APIs Setiap layanan mikro dalam arsitektur ini dapat diperbarui, digunakan, dan diskalakan untuk memenuhi permintaan fungsi tertentu dari suatu aplikasi. Untuk informasi selengkapnya, lihat [Menerapkan layanan mikro di AWS](#).

## Program Percepatan Migrasi (MAP)

AWS Program yang menyediakan dukungan konsultasi, pelatihan, dan layanan untuk membantu organisasi membangun fondasi operasional yang kuat untuk pindah ke cloud, dan untuk membantu mengimbangi biaya awal migrasi. MAP mencakup metodologi migrasi untuk mengeksekusi migrasi lama dengan cara metodis dan seperangkat alat untuk mengotomatisasi dan mempercepat skenario migrasi umum.

## migrasi dalam skala

Proses memindahkan sebagian besar portofolio aplikasi ke cloud dalam gelombang, dengan lebih banyak aplikasi bergerak pada tingkat yang lebih cepat di setiap gelombang. Fase ini menggunakan praktik dan pelajaran terbaik dari fase sebelumnya untuk mengimplementasikan pabrik migrasi tim, alat, dan proses untuk merampingkan migrasi beban kerja melalui otomatisasi dan pengiriman tangkas. Ini adalah fase ketiga dari [strategi AWS migrasi](#).

## pabrik migrasi

Tim lintas fungsi yang merampingkan migrasi beban kerja melalui pendekatan otomatis dan gesit. Tim pabrik migrasi biasanya mencakup operasi, analis dan pemilik bisnis, insinyur migrasi, pengembang, dan DevOps profesional yang bekerja di sprint. Antara 20 dan 50 persen portofolio aplikasi perusahaan terdiri dari pola berulang yang dapat dioptimalkan dengan pendekatan pabrik. Untuk informasi selengkapnya, lihat [diskusi tentang pabrik migrasi](#) dan [panduan Pabrik Migrasi Cloud](#) di kumpulan konten ini.

## metadata migrasi

Informasi tentang aplikasi dan server yang diperlukan untuk menyelesaikan migrasi. Setiap pola migrasi memerlukan satu set metadata migrasi yang berbeda. Contoh metadata migrasi termasuk subnet target, grup keamanan, dan akun. AWS

## pola migrasi

Tugas migrasi berulang yang merinci strategi migrasi, tujuan migrasi, dan aplikasi atau layanan migrasi yang digunakan. Contoh: Rehost migrasi ke Amazon EC2 AWS dengan Layanan Migrasi Aplikasi.

## Penilaian Portofolio Migrasi (MPA)

Alat online yang menyediakan informasi untuk memvalidasi kasus bisnis untuk bermigrasi ke. AWS Cloud MPA menyediakan penilaian portofolio terperinci (ukuran kanan server, harga, perbandingan TCO, analisis biaya migrasi) serta perencanaan migrasi (analisis data aplikasi dan pengumpulan data, pengelompokan aplikasi, prioritas migrasi, dan perencanaan gelombang). [Alat MPA](#) (memerlukan login) tersedia gratis untuk semua AWS konsultan dan konsultan APN Partner.

## Penilaian Kesiapan Migrasi (MRA)

Proses mendapatkan wawasan tentang status kesiapan cloud organisasi, mengidentifikasi kekuatan dan kelemahan, dan membangun rencana aksi untuk menutup kesenjangan yang diidentifikasi, menggunakan CAF. AWS Untuk informasi selengkapnya, lihat [panduan kesiapan migrasi](#). MRA adalah tahap pertama dari [strategi AWS migrasi](#).

## strategi migrasi

Pendekatan yang digunakan untuk memigrasikan beban kerja ke. AWS Cloud Untuk informasi lebih lanjut, lihat entri [7 Rs](#) di glosarium ini dan lihat [Memobilisasi organisasi Anda untuk mempercepat](#) migrasi skala besar.

## ML

Lihat [pembelajaran mesin](#).

## modernisasi

Mengubah aplikasi usang (warisan atau monolitik) dan infrastrukturnya menjadi sistem yang gesit, elastis, dan sangat tersedia di cloud untuk mengurangi biaya, mendapatkan efisiensi, dan memanfaatkan inovasi. Untuk informasi selengkapnya, lihat [Strategi untuk memodernisasi aplikasi di](#). AWS Cloud

## penilaian kesiapan modernisasi

Evaluasi yang membantu menentukan kesiapan modernisasi aplikasi organisasi; mengidentifikasi manfaat, risiko, dan dependensi; dan menentukan seberapa baik organisasi dapat mendukung keadaan masa depan aplikasi tersebut. Hasil penilaian adalah cetak biru arsitektur target, peta jalan yang merinci fase pengembangan dan tonggak untuk proses modernisasi, dan rencana aksi untuk mengatasi kesenjangan yang diidentifikasi. Untuk informasi lebih lanjut, lihat [Mengevaluasi kesiapan modernisasi untuk](#) aplikasi di. AWS Cloud

## aplikasi monolitik (monolit)

Aplikasi yang berjalan sebagai layanan tunggal dengan proses yang digabungkan secara ketat. Aplikasi monolitik memiliki beberapa kelemahan. Jika satu fitur aplikasi mengalami lonjakan permintaan, seluruh arsitektur harus diskalakan. Menambahkan atau meningkatkan fitur aplikasi monolitik juga menjadi lebih kompleks ketika basis kode tumbuh. Untuk mengatasi masalah ini, Anda dapat menggunakan arsitektur microservices. Untuk informasi lebih lanjut, lihat [Mengurai monolit](#) menjadi layanan mikro.

## MPA

Lihat [Penilaian Portofolio Migrasi](#).

## MQTT

Lihat [Transportasi Telemetri Antrian Pesan](#).

## klasifikasi multiclass

Sebuah proses yang membantu menghasilkan prediksi untuk beberapa kelas (memprediksi satu dari lebih dari dua hasil). Misalnya, model ML mungkin bertanya “Apakah produk ini buku, mobil, atau telepon?” atau “Kategori produk mana yang paling menarik bagi pelanggan ini?”

## infrastruktur yang bisa berubah

Model yang memperbarui dan memodifikasi infrastruktur yang ada untuk beban kerja produksi. Untuk meningkatkan konsistensi, keandalan, dan prediktabilitas, AWS Well-Architected Framework merekomendasikan penggunaan infrastruktur yang [tidak](#) dapat diubah sebagai praktik terbaik.

## O

### OAC

Lihat [kontrol akses asal](#).

### OAI

Lihat [identitas akses asal](#).

### OCM

Lihat [manajemen perubahan organisasi](#).

### migrasi offline

Metode migrasi di mana beban kerja sumber diturunkan selama proses migrasi. Metode ini melibatkan waktu henti yang diperpanjang dan biasanya digunakan untuk beban kerja kecil dan tidak kritis.

### OI

Lihat [integrasi operasi](#).

### OLA

Lihat [perjanjian tingkat operasional](#).

### migrasi online

Metode migrasi di mana beban kerja sumber disalin ke sistem target tanpa diambil offline. Aplikasi yang terhubung ke beban kerja dapat terus berfungsi selama migrasi. Metode ini melibatkan waktu henti nol hingga minimal dan biasanya digunakan untuk beban kerja produksi yang kritis.

### OPC-UA

Lihat [Komunikasi Proses Terbuka - Arsitektur Terpadu](#).

### Komunikasi Proses Terbuka - Arsitektur Terpadu (OPC-UA)

Protokol komunikasi machine-to-machine (M2M) untuk otomasi industri. OPC-UA menyediakan standar interoperabilitas dengan enkripsi data, otentikasi, dan skema otorisasi.

### perjanjian tingkat operasional (OLA)

Perjanjian yang menjelaskan apa yang dijanjikan kelompok TI fungsional untuk diberikan satu sama lain, untuk mendukung perjanjian tingkat layanan (SLA).

## Tinjauan Kesiapan Operasional (ORR)

Daftar pertanyaan dan praktik terbaik terkait yang membantu Anda memahami, mengevaluasi, mencegah, atau mengurangi ruang lingkup insiden dan kemungkinan kegagalan. Untuk informasi lebih lanjut, lihat [Ulasan Kesiapan Operasional \(ORR\)](#) dalam Kerangka Kerja Well-Architected AWS .

## teknologi operasional (OT)

Sistem perangkat keras dan perangkat lunak yang bekerja dengan lingkungan fisik untuk mengendalikan operasi industri, peralatan, dan infrastruktur. Di bidang manufaktur, integrasi sistem OT dan teknologi informasi (TI) adalah fokus utama untuk transformasi [Industri 4.0](#).

## integrasi operasi (OI)

Proses modernisasi operasi di cloud, yang melibatkan perencanaan kesiapan, otomatisasi, dan integrasi. Untuk informasi selengkapnya, lihat [panduan integrasi operasi](#).

## jejak organisasi

Jejak yang dibuat oleh AWS CloudTrail itu mencatat semua peristiwa untuk semua Akun AWS dalam organisasi di AWS Organizations. Jejak ini dibuat di setiap Akun AWS bagian organisasi dan melacak aktivitas di setiap akun. Untuk informasi selengkapnya, lihat [Membuat jejak untuk organisasi](#) dalam CloudTrail dokumentasi.

## manajemen perubahan organisasi (OCM)

Kerangka kerja untuk mengelola transformasi bisnis utama yang mengganggu dari perspektif orang, budaya, dan kepemimpinan. OCM membantu organisasi mempersiapkan, dan transisi ke, sistem dan strategi baru dengan mempercepat adopsi perubahan, mengatasi masalah transisi, dan mendorong perubahan budaya dan organisasi. Dalam strategi AWS migrasi, kerangka kerja ini disebut percepatan orang, karena kecepatan perubahan yang diperlukan dalam proyek adopsi cloud. Untuk informasi lebih lanjut, lihat [panduan OCM](#).

## kontrol akses asal (OAC)

Di CloudFront, opsi yang disempurnakan untuk membatasi akses untuk mengamankan konten Amazon Simple Storage Service (Amazon S3) Anda. OAC mendukung semua bucket S3 di semua Wilayah AWS, enkripsi sisi server dengan AWS KMS (SSE-KMS), dan dinamis dan permintaan ke bucket S3. PUT DELETE

## identitas akses asal (OAI)

Di CloudFront, opsi untuk membatasi akses untuk mengamankan konten Amazon S3 Anda. Saat Anda menggunakan OAI, CloudFront buat prinsipal yang dapat diautentikasi oleh Amazon S3. Prinsipal yang diautentikasi dapat mengakses konten dalam bucket S3 hanya melalui distribusi tertentu. CloudFront Lihat juga [OAC](#), yang menyediakan kontrol akses yang lebih terperinci dan ditingkatkan.

## ORR

Lihat [tinjauan kesiapan operasional](#).

## OT

Lihat [teknologi operasional](#).

## keluar (jalan keluar) VPC

Dalam arsitektur AWS multi-akun, VPC yang menangani koneksi jaringan yang dimulai dari dalam aplikasi. [Arsitektur Referensi AWS Keamanan](#) merekomendasikan pengaturan akun Jaringan Anda dengan inbound, outbound, dan inspeksi VPCs untuk melindungi antarmuka dua arah antara aplikasi Anda dan internet yang lebih luas.

## P

### batas izin

Kebijakan manajemen IAM yang dilampirkan pada prinsipal IAM untuk menetapkan izin maksimum yang dapat dimiliki pengguna atau peran. Untuk informasi selengkapnya, lihat [Batas izin](#) dalam dokumentasi IAM.

### Informasi Identifikasi Pribadi (PII)

Informasi yang, jika dilihat secara langsung atau dipasangkan dengan data terkait lainnya, dapat digunakan untuk menyimpulkan identitas individu secara wajar. Contoh PII termasuk nama, alamat, dan informasi kontak.

## PII

Lihat informasi yang [dapat diidentifikasi secara pribadi](#).

## buku pedoman

Serangkaian langkah yang telah ditentukan sebelumnya yang menangkap pekerjaan yang terkait dengan migrasi, seperti mengirimkan fungsi operasi inti di cloud. Buku pedoman dapat berupa skrip, runbook otomatis, atau ringkasan proses atau langkah-langkah yang diperlukan untuk mengoperasikan lingkungan modern Anda.

## PLC

Lihat [pengontrol logika yang dapat diprogram](#).

## PLM

Lihat [manajemen siklus hidup produk](#).

## kebijakan

[Objek yang dapat menentukan izin \(lihat kebijakan berbasis identitas\), menentukan kondisi akses \(lihat kebijakan berbasis sumber daya\), atau menentukan izin maksimum untuk semua akun di organisasi \(lihat kebijakan kontrol layanan\). AWS Organizations](#)

## ketekunan poliglott

Secara independen memilih teknologi penyimpanan data microservice berdasarkan pola akses data dan persyaratan lainnya. Jika layanan mikro Anda memiliki teknologi penyimpanan data yang sama, mereka dapat menghadapi tantangan implementasi atau mengalami kinerja yang buruk. Layanan mikro lebih mudah diimplementasikan dan mencapai kinerja dan skalabilitas yang lebih baik jika mereka menggunakan penyimpanan data yang paling sesuai dengan kebutuhan mereka.

## penilaian portofolio

Proses menemukan, menganalisis, dan memprioritaskan portofolio aplikasi untuk merencanakan migrasi. Untuk informasi selengkapnya, lihat [Mengevaluasi kesiapan migrasi](#).

## predikat

Kondisi kueri yang mengembalikan `true` atau `false`, biasanya terletak di `WHERE` klausa.

## predikat pushdown

Teknik pengoptimalan kueri database yang menyaring data dalam kueri sebelum transfer. Ini mengurangi jumlah data yang harus diambil dan diproses dari database relasional, dan meningkatkan kinerja kueri.

## kontrol preventif

Kontrol keamanan yang dirancang untuk mencegah suatu peristiwa terjadi. Kontrol ini adalah garis pertahanan pertama untuk membantu mencegah akses tidak sah atau perubahan yang tidak diinginkan ke jaringan Anda. Untuk informasi selengkapnya, lihat [Kontrol pencegahan dalam Menerapkan kontrol](#) keamanan pada AWS.

## principal

Entitas AWS yang dapat melakukan tindakan dan mengakses sumber daya. Entitas ini biasanya merupakan pengguna root untuk Akun AWS, peran IAM, atau pengguna. Untuk informasi selengkapnya, lihat Prinsip dalam [istilah dan konsep Peran](#) dalam dokumentasi IAM.

## privasi berdasarkan desain

Pendekatan rekayasa sistem yang memperhitungkan privasi melalui seluruh proses pengembangan.

## zona yang dihosting pribadi

Container yang menyimpan informasi tentang bagaimana Anda ingin Amazon Route 53 merespons kueri DNS untuk domain dan subdomainnya dalam satu atau lebih VPCs. Untuk informasi selengkapnya, lihat [Bekerja dengan zona yang dihosting pribadi](#) di dokumentasi Route 53.

## kontrol proaktif

[Kontrol keamanan](#) yang dirancang untuk mencegah penyebaran sumber daya yang tidak sesuai. Kontrol ini memindai sumber daya sebelum disediakan. Jika sumber daya tidak sesuai dengan kontrol, maka itu tidak disediakan. Untuk informasi selengkapnya, lihat [panduan referensi Kontrol](#) dalam AWS Control Tower dokumentasi dan lihat [Kontrol proaktif](#) dalam Menerapkan kontrol keamanan pada AWS.

## manajemen siklus hidup produk (PLM)

Manajemen data dan proses untuk suatu produk di seluruh siklus hidupnya, mulai dari desain, pengembangan, dan peluncuran, melalui pertumbuhan dan kematangan, hingga penurunan dan penghapusan.

## lingkungan produksi

Lihat [lingkungan](#).

## pengontrol logika yang dapat diprogram (PLC)

Di bidang manufaktur, komputer yang sangat andal dan mudah beradaptasi yang memantau mesin dan mengotomatiskan proses manufaktur.

## rantai cepat

Menggunakan output dari satu prompt [LLM](#) sebagai input untuk prompt berikutnya untuk menghasilkan respons yang lebih baik. Teknik ini digunakan untuk memecah tugas yang kompleks menjadi subtugas, atau untuk secara iteratif memperbaiki atau memperluas respons awal. Ini membantu meningkatkan akurasi dan relevansi respons model dan memungkinkan hasil yang lebih terperinci dan dipersonalisasi.

## pseudonimisasi

Proses penggantian pengidentifikasi pribadi dalam kumpulan data dengan nilai placeholder. Pseudonimisasi dapat membantu melindungi privasi pribadi. Data pseudonim masih dianggap sebagai data pribadi.

## publish/subscribe (pub/sub)

Pola yang memungkinkan komunikasi asinkron antara layanan mikro untuk meningkatkan skalabilitas dan daya tanggap. Misalnya, dalam [MES](#) berbasis layanan mikro, layanan mikro dapat mempublikasikan pesan peristiwa ke saluran yang dapat berlangganan layanan mikro lainnya. Sistem dapat menambahkan layanan mikro baru tanpa mengubah layanan penerbitan.

# Q

## rencana kueri

Serangkaian langkah, seperti instruksi, yang digunakan untuk mengakses data dalam sistem database relasional SQL.

## regresi rencana kueri

Ketika pengoptimal layanan database memilih rencana yang kurang optimal daripada sebelum perubahan yang diberikan ke lingkungan database. Hal ini dapat disebabkan oleh perubahan statistik, kendala, pengaturan lingkungan, pengikatan parameter kueri, dan pembaruan ke mesin database.

# R

## Matriks RACI

Lihat [bertanggung jawab, akuntabel, dikonsultasikan, diinformasikan \(RACI\)](#).

## LAP

Lihat [Retrieval Augmented Generation](#).

## ransomware

Perangkat lunak berbahaya yang dirancang untuk memblokir akses ke sistem komputer atau data sampai pembayaran dilakukan.

## Matriks RASCI

Lihat [bertanggung jawab, akuntabel, dikonsultasikan, diinformasikan \(RACI\)](#).

## RCAC

Lihat [kontrol akses baris dan kolom](#).

## replika baca

Salinan database yang digunakan untuk tujuan read-only. Anda dapat merutekan kueri ke replika baca untuk mengurangi beban pada database utama Anda.

## arsitek ulang

Lihat [7 Rs](#).

## tujuan titik pemulihan (RPO)

Jumlah waktu maksimum yang dapat diterima sejak titik pemulihan data terakhir. Ini menentukan apa yang dianggap sebagai kehilangan data yang dapat diterima antara titik pemulihan terakhir dan gangguan layanan.

## tujuan waktu pemulihan (RTO)

Penundaan maksimum yang dapat diterima antara gangguan layanan dan pemulihan layanan.

## refactor

Lihat [7 Rs](#).

## Region

Kumpulan AWS sumber daya di wilayah geografis. Masing-masing AWS Region terisolasi dan independen dari yang lain untuk memberikan toleransi kesalahan, stabilitas, dan ketahanan.

Untuk informasi selengkapnya, lihat [Menentukan Wilayah AWS akun yang dapat digunakan](#).

## regresi

Teknik ML yang memprediksi nilai numerik. Misalnya, untuk memecahkan masalah “Berapa harga rumah ini akan dijual?” Model ML dapat menggunakan model regresi linier untuk memprediksi harga jual rumah berdasarkan fakta yang diketahui tentang rumah (misalnya, luas persegi).

## rehost

Lihat [7 Rs](#).

## melepaskan

Dalam proses penyebaran, tindakan mempromosikan perubahan pada lingkungan produksi.

## memindahkan

Lihat [7 Rs](#).

## memplatform ulang

Lihat [7 Rs](#).

## pembelian kembali

Lihat [7 Rs](#).

## ketahanan

Kemampuan aplikasi untuk melawan atau pulih dari gangguan. [Ketersediaan tinggi](#) dan [pemulihan bencana](#) adalah pertimbangan umum ketika merencanakan ketahanan di AWS Cloud.

Untuk informasi lebih lanjut, lihat [AWS Cloud Ketahanan](#).

## kebijakan berbasis sumber daya

Kebijakan yang dilampirkan ke sumber daya, seperti bucket Amazon S3, titik akhir, atau kunci enkripsi. Jenis kebijakan ini menentukan prinsipal mana yang diizinkan mengakses, tindakan yang didukung, dan kondisi lain yang harus dipenuhi.

## matriks yang bertanggung jawab, akuntabel, dikonsultasikan, diinformasikan (RACI)

Matriks yang mendefinisikan peran dan tanggung jawab untuk semua pihak yang terlibat dalam kegiatan migrasi dan operasi cloud. Nama matriks berasal dari jenis tanggung jawab yang

didefinisikan dalam matriks: bertanggung jawab (R), akuntabel (A), dikonsultasikan (C), dan diinformasikan (I). Tipe dukungan (S) adalah opsional. Jika Anda menyertakan dukungan, matriks disebut matriks RASCI, dan jika Anda mengecualikannya, itu disebut matriks RACI.

### kontrol responsif

Kontrol keamanan yang dirancang untuk mendorong remediasi efek samping atau penyimpangan dari garis dasar keamanan Anda. Untuk informasi selengkapnya, lihat [Kontrol responsif](#) dalam Menerapkan kontrol keamanan pada AWS.

### melestarikan

Lihat [7 Rs](#).

### pensiun

Lihat [7 Rs](#).

### Retrieval Augmented Generation (RAG)

Teknologi [AI generatif](#) di mana [LLM](#) merujuk sumber data otoritatif yang berada di luar sumber data pelatihannya sebelum menghasilkan respons. Misalnya, model RAG mungkin melakukan pencarian semantik dari basis pengetahuan organisasi atau data kustom. Untuk informasi lebih lanjut, lihat [Apa itu RAG](#).

### rotasi

Proses memperbarui [rahasia](#) secara berkala untuk membuatnya lebih sulit bagi penyerang untuk mengakses kredensial.

### kontrol akses baris dan kolom (RCAC)

Penggunaan ekspresi SQL dasar dan fleksibel yang telah menetapkan aturan akses. RCAC terdiri dari izin baris dan topeng kolom.

### RPO

Lihat [tujuan titik pemulihan](#).

### RTO

Lihat [tujuan waktu pemulihan](#).

## buku runbook

Satu set prosedur manual atau otomatis yang diperlukan untuk melakukan tugas tertentu. Ini biasanya dibangun untuk merampingkan operasi berulang atau prosedur dengan tingkat kesalahan yang tinggi.

## D

### SAML 2.0

Standar terbuka yang digunakan oleh banyak penyedia identitas (IdPs). Fitur ini memungkinkan sistem masuk tunggal gabungan (SSO), sehingga pengguna dapat masuk ke Konsol Manajemen AWS atau memanggil operasi AWS API tanpa Anda harus membuat pengguna di IAM untuk semua orang di organisasi Anda. Untuk informasi lebih lanjut tentang federasi berbasis SAMP 2.0, lihat [Tentang federasi berbasis SAMP 2.0](#) dalam dokumentasi IAM.

### SCADA

Lihat [kontrol pengawasan dan akuisisi data](#).

### SCP

Lihat [kebijakan kontrol layanan](#).

### Rahasia

Dalam AWS Secrets Manager, informasi rahasia atau terbatas, seperti kata sandi atau kredensial pengguna, yang Anda simpan dalam bentuk terenkripsi. Ini terdiri dari nilai rahasia dan metadatanya. Nilai rahasia dapat berupa biner, string tunggal, atau beberapa string. Untuk informasi selengkapnya, lihat [Apa yang ada di rahasia Secrets Manager?](#) dalam dokumentasi Secrets Manager.

### keamanan dengan desain

Pendekatan rekayasa sistem yang memperhitungkan keamanan melalui seluruh proses pengembangan.

### kontrol keamanan

Pagar pembatas teknis atau administratif yang mencegah, mendeteksi, atau mengurangi kemampuan pelaku ancaman untuk mengeksploitasi kerentanan keamanan. [Ada empat jenis kontrol keamanan utama: preventif, detektif, responsif, dan proaktif.](#)

## pengerasan keamanan

Proses mengurangi permukaan serangan untuk membuatnya lebih tahan terhadap serangan. Ini dapat mencakup tindakan seperti menghapus sumber daya yang tidak lagi diperlukan, menerapkan praktik keamanan terbaik untuk memberikan hak istimewa paling sedikit, atau menonaktifkan fitur yang tidak perlu dalam file konfigurasi.

## sistem informasi keamanan dan manajemen acara (SIEM)

Alat dan layanan yang menggabungkan sistem manajemen informasi keamanan (SIM) dan manajemen acara keamanan (SEM). Sistem SIEM mengumpulkan, memantau, dan menganalisis data dari server, jaringan, perangkat, dan sumber lain untuk mendeteksi ancaman dan pelanggaran keamanan, dan untuk menghasilkan peringatan.

## otomatisasi respons keamanan

Tindakan yang telah ditentukan dan diprogram yang dirancang untuk secara otomatis merespons atau memulihkan peristiwa keamanan. Otomatisasi ini berfungsi sebagai kontrol keamanan [detektif](#) atau [responsif](#) yang membantu Anda menerapkan praktik terbaik AWS keamanan. Contoh tindakan respons otomatis termasuk memodifikasi grup keamanan VPC, menambal instans Amazon EC2, atau memutar kredensial.

## enkripsi sisi server

Enkripsi data di tujuannya, oleh Layanan AWS yang menerimanya.

## kebijakan kontrol layanan (SCP)

Kebijakan yang menyediakan kontrol terpusat atas izin untuk semua akun di organisasi. AWS Organizations SCPs menentukan pagar pembatas atau menetapkan batasan pada tindakan yang dapat didelegasikan oleh administrator kepada pengguna atau peran. Anda dapat menggunakan SCPs daftar izin atau daftar penolakan, untuk menentukan layanan atau tindakan mana yang diizinkan atau dilarang. Untuk informasi selengkapnya, lihat [Kebijakan kontrol layanan](#) dalam AWS Organizations dokumentasi.

## titik akhir layanan

URL titik masuk untuk file Layanan AWS. Anda dapat menggunakan endpoint untuk terhubung secara terprogram ke layanan target. Untuk informasi selengkapnya, lihat [Layanan AWS titik akhir](#) di Referensi Umum AWS.

## perjanjian tingkat layanan (SLA)

Perjanjian yang menjelaskan apa yang dijanjikan tim TI untuk diberikan kepada pelanggan mereka, seperti waktu kerja dan kinerja layanan.

## indikator tingkat layanan (SLI)

Pengukuran aspek kinerja layanan, seperti tingkat kesalahan, ketersediaan, atau throughputnya.

## tujuan tingkat layanan (SLO)

Metrik target yang mewakili kesehatan layanan, yang diukur dengan indikator [tingkat layanan](#).

## model tanggung jawab bersama

Model yang menjelaskan tanggung jawab yang Anda bagikan AWS untuk keamanan dan kepatuhan cloud. AWS bertanggung jawab atas keamanan cloud, sedangkan Anda bertanggung jawab atas keamanan di cloud. Untuk informasi selengkapnya, lihat [Model tanggung jawab bersama](#).

## SIEM

Lihat [informasi keamanan dan sistem manajemen acara](#).

## titik kegagalan tunggal (SPOF)

Kegagalan dalam satu komponen penting dari aplikasi yang dapat mengganggu sistem.

## SLA

Lihat [perjanjian tingkat layanan](#).

## SLI

Lihat [indikator tingkat layanan](#).

## SLO

Lihat [tujuan tingkat layanan](#).

## split-and-seed model

Pola untuk menskalakan dan mempercepat proyek modernisasi. Ketika fitur baru dan rilis produk didefinisikan, tim inti berpisah untuk membuat tim produk baru. Ini membantu meningkatkan kemampuan dan layanan organisasi Anda, meningkatkan produktivitas pengembang, dan

mendukung inovasi yang cepat. Untuk informasi lebih lanjut, lihat [Pendekatan bertahap untuk memodernisasi aplikasi](#) di AWS Cloud

## SPOF

Lihat [satu titik kegagalan](#).

## skema bintang

Struktur organisasi database yang menggunakan satu tabel fakta besar untuk menyimpan data transaksional atau terukur dan menggunakan satu atau lebih tabel dimensi yang lebih kecil untuk menyimpan atribut data. Struktur ini dirancang untuk digunakan dalam [gudang data](#) atau untuk tujuan intelijen bisnis.

## pola ara pencekik

Pendekatan untuk memodernisasi sistem monolitik dengan menulis ulang secara bertahap dan mengganti fungsionalitas sistem sampai sistem warisan dapat dinonaktifkan. Pola ini menggunakan analogi pohon ara yang tumbuh menjadi pohon yang sudah mapan dan akhirnya mengatasi dan menggantikan inangnya. Pola ini [diperkenalkan oleh Martin Fowler](#) sebagai cara untuk mengelola risiko saat menulis ulang sistem monolitik. Untuk contoh cara menerapkan pola ini, lihat [Memodernisasi layanan web Microsoft ASP.NET \(ASMX\) lama secara bertahap menggunakan container dan Amazon API Gateway](#).

## subnet

Rentang alamat IP dalam VPC Anda. Subnet harus berada di Availability Zone tunggal.

## kontrol pengawasan dan akuisisi data (SCADA)

Di bidang manufaktur, sistem yang menggunakan perangkat keras dan perangkat lunak untuk memantau aset fisik dan operasi produksi.

## enkripsi simetris

Algoritma enkripsi yang menggunakan kunci yang sama untuk mengenkripsi dan mendekripsi data.

## pengujian sintetis

Menguji sistem dengan cara yang mensimulasikan interaksi pengguna untuk mendeteksi potensi masalah atau untuk memantau kinerja. Anda dapat menggunakan [Amazon CloudWatch Synthetics](#) untuk membuat tes ini.

## sistem prompt

Teknik untuk memberikan konteks, instruksi, atau pedoman ke [LLM](#) untuk mengarahkan perilakunya. Permintaan sistem membantu mengatur konteks dan menetapkan aturan untuk interaksi dengan pengguna.

## T

### tag

Pasangan nilai kunci yang bertindak sebagai metadata untuk mengatur sumber daya Anda. AWS Tanda membantu Anda mengelola, mengidentifikasi, mengatur, dan memfilter sumber daya. Untuk informasi selengkapnya, lihat [Menandai AWS sumber daya Anda](#).

### variabel target

Nilai yang Anda coba prediksi dalam ML yang diawasi. Ini juga disebut sebagai variabel hasil. Misalnya, dalam pengaturan manufaktur, variabel target bisa menjadi cacat produk.

### daftar tugas

Alat yang digunakan untuk melacak kemajuan melalui runbook. Daftar tugas berisi ikhtisar runbook dan daftar tugas umum yang harus diselesaikan. Untuk setiap tugas umum, itu termasuk perkiraan jumlah waktu yang dibutuhkan, pemilik, dan kemajuan.

### lingkungan uji

Lihat [lingkungan](#).

### pelatihan

Untuk menyediakan data bagi model ML Anda untuk dipelajari. Data pelatihan harus berisi jawaban yang benar. Algoritma pembelajaran menemukan pola dalam data pelatihan yang memetakan atribut data input ke target (jawaban yang ingin Anda prediksi). Ini menghasilkan model ML yang menangkap pola-pola ini. Anda kemudian dapat menggunakan model ML untuk membuat prediksi pada data baru yang Anda tidak tahu targetnya.

### gerbang transit

Hub transit jaringan yang dapat Anda gunakan untuk menghubungkan jaringan Anda VPCs dan lokal. Untuk informasi selengkapnya, lihat [Apa itu gateway transit](#) dalam AWS Transit Gateway dokumentasi.

## alur kerja berbasis batang

Pendekatan di mana pengembang membangun dan menguji fitur secara lokal di cabang fitur dan kemudian menggabungkan perubahan tersebut ke cabang utama. Cabang utama kemudian dibangun untuk pengembangan, praproduksi, dan lingkungan produksi, secara berurutan.

## akses tepercaya

Memberikan izin ke layanan yang Anda tentukan untuk melakukan tugas di organisasi Anda di dalam AWS Organizations dan di akunnya atas nama Anda. Layanan tepercaya menciptakan peran terkait layanan di setiap akun, ketika peran itu diperlukan, untuk melakukan tugas manajemen untuk Anda. Untuk informasi selengkapnya, lihat [Menggunakan AWS Organizations dengan AWS layanan lain](#) dalam AWS Organizations dokumentasi.

## penyetelan

Untuk mengubah aspek proses pelatihan Anda untuk meningkatkan akurasi model ML. Misalnya, Anda dapat melatih model ML dengan membuat set pelabelan, menambahkan label, dan kemudian mengulangi langkah-langkah ini beberapa kali di bawah pengaturan yang berbeda untuk mengoptimalkan model.

## tim dua pizza

Sebuah DevOps tim kecil yang bisa Anda beri makan dengan dua pizza. Ukuran tim dua pizza memastikan peluang terbaik untuk berkolaborasi dalam pengembangan perangkat lunak.

# U

## waswas

Sebuah konsep yang mengacu pada informasi yang tidak tepat, tidak lengkap, atau tidak diketahui yang dapat merusak keandalan model ML prediktif. Ada dua jenis ketidakpastian: ketidakpastian epistemik disebabkan oleh data yang terbatas dan tidak lengkap, sedangkan ketidakpastian aleatorik disebabkan oleh kebisingan dan keacakan yang melekat dalam data. Untuk informasi lebih lanjut, lihat panduan [Mengukur ketidakpastian dalam sistem pembelajaran mendalam](#).

## tugas yang tidak terdiferensiasi

Juga dikenal sebagai angkat berat, pekerjaan yang diperlukan untuk membuat dan mengoperasikan aplikasi tetapi itu tidak memberikan nilai langsung kepada pengguna akhir atau

memberikan keunggulan kompetitif. Contoh tugas yang tidak terdiferensiasi termasuk pengadaan, pemeliharaan, dan perencanaan kapasitas.

lingkungan atas

Lihat [lingkungan](#).

## V

menyedot debu

Operasi pemeliharaan database yang melibatkan pembersihan setelah pembaruan tambahan untuk merebut kembali penyimpanan dan meningkatkan kinerja.

kendali versi

Proses dan alat yang melacak perubahan, seperti perubahan kode sumber dalam repositori.

Peering VPC

Koneksi antara dua VPCs yang memungkinkan Anda untuk merutekan lalu lintas dengan menggunakan alamat IP pribadi. Untuk informasi selengkapnya, lihat [Apa itu peering VPC](#) di dokumentasi VPC Amazon.

kerentanan

Kelemahan perangkat lunak atau perangkat keras yang membahayakan keamanan sistem.

## W

cache hangat

Cache buffer yang berisi data terkini dan relevan yang sering diakses. Instance database dapat membaca dari cache buffer, yang lebih cepat daripada membaca dari memori utama atau disk.

data hangat

Data yang jarang diakses. Saat menanyakan jenis data ini, kueri yang cukup lambat biasanya dapat diterima.

## fungsi jendela

Fungsi SQL yang melakukan perhitungan pada sekelompok baris yang berhubungan dengan catatan saat ini. Fungsi jendela berguna untuk memproses tugas, seperti menghitung rata-rata bergerak atau mengakses nilai baris berdasarkan posisi relatif dari baris saat ini.

## beban kerja

Kumpulan sumber daya dan kode yang memberikan nilai bisnis, seperti aplikasi yang dihadapi pelanggan atau proses backend.

## aliran kerja

Grup fungsional dalam proyek migrasi yang bertanggung jawab atas serangkaian tugas tertentu. Setiap alur kerja independen tetapi mendukung alur kerja lain dalam proyek. Misalnya, alur kerja portofolio bertanggung jawab untuk memprioritaskan aplikasi, perencanaan gelombang, dan mengumpulkan metadata migrasi. Alur kerja portofolio mengirimkan aset ini ke alur kerja migrasi, yang kemudian memigrasikan server dan aplikasi.

## CACING

Lihat [menulis sekali, baca banyak](#).

## WQF

Lihat [AWS Kerangka Kualifikasi Beban Kerja](#).

## tulis sekali, baca banyak (WORM)

Model penyimpanan yang menulis data satu kali dan mencegah data dihapus atau dimodifikasi. Pengguna yang berwenang dapat membaca data sebanyak yang diperlukan, tetapi mereka tidak dapat mengubahnya. Infrastruktur penyimpanan data ini dianggap [tidak dapat diubah](#).

## Z

### eksploitasi zero-day

Serangan, biasanya malware, yang memanfaatkan kerentanan [zero-day](#).

### kerentanan zero-day

Cacat atau kerentanan yang tak tanggung-tanggung dalam sistem produksi. Aktor ancaman dapat menggunakan jenis kerentanan ini untuk menyerang sistem. Pengembang sering menyadari kerentanan sebagai akibat dari serangan tersebut.

---

## bisikan zero-shot

Memberikan [LLM](#) dengan instruksi untuk melakukan tugas tetapi tidak ada contoh (tembak) yang dapat membantu membimbingnya. LLM harus menggunakan pengetahuan pra-terlatih untuk menangani tugas. Efektivitas bidikan nol tergantung pada kompleksitas tugas dan kualitas prompt. Lihat juga beberapa [bidikan yang diminta](#).

## aplikasi zombie

Aplikasi yang memiliki CPU rata-rata dan penggunaan memori di bawah 5 persen. Dalam proyek migrasi, adalah umum untuk menghentikan aplikasi ini.

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.