Architektur-Diagramme

Parallele API-Zusammensetzung in AWS



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

Parallele API-Zusammensetzung in AWS: Architektur-Diagramme

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

Die Handelsmarken und die Handelsaufmachung von Amazon dürfen nicht in einer Weise in Verbindung mit nicht von Amazon stammenden Produkten oder Services verwendet werden, durch die Kunden irregeführt werden könnten oder Amazon in schlechtem Licht dargestellt oder diskreditiert werden könnte. Alle anderen Handelsmarken, die nicht Eigentum von Amazon sind, gehören den jeweiligen Besitzern, die möglicherweise zu Amazon gehören oder nicht, mit Amazon verbunden sind oder von Amazon gesponsert werden.

Table of Contents

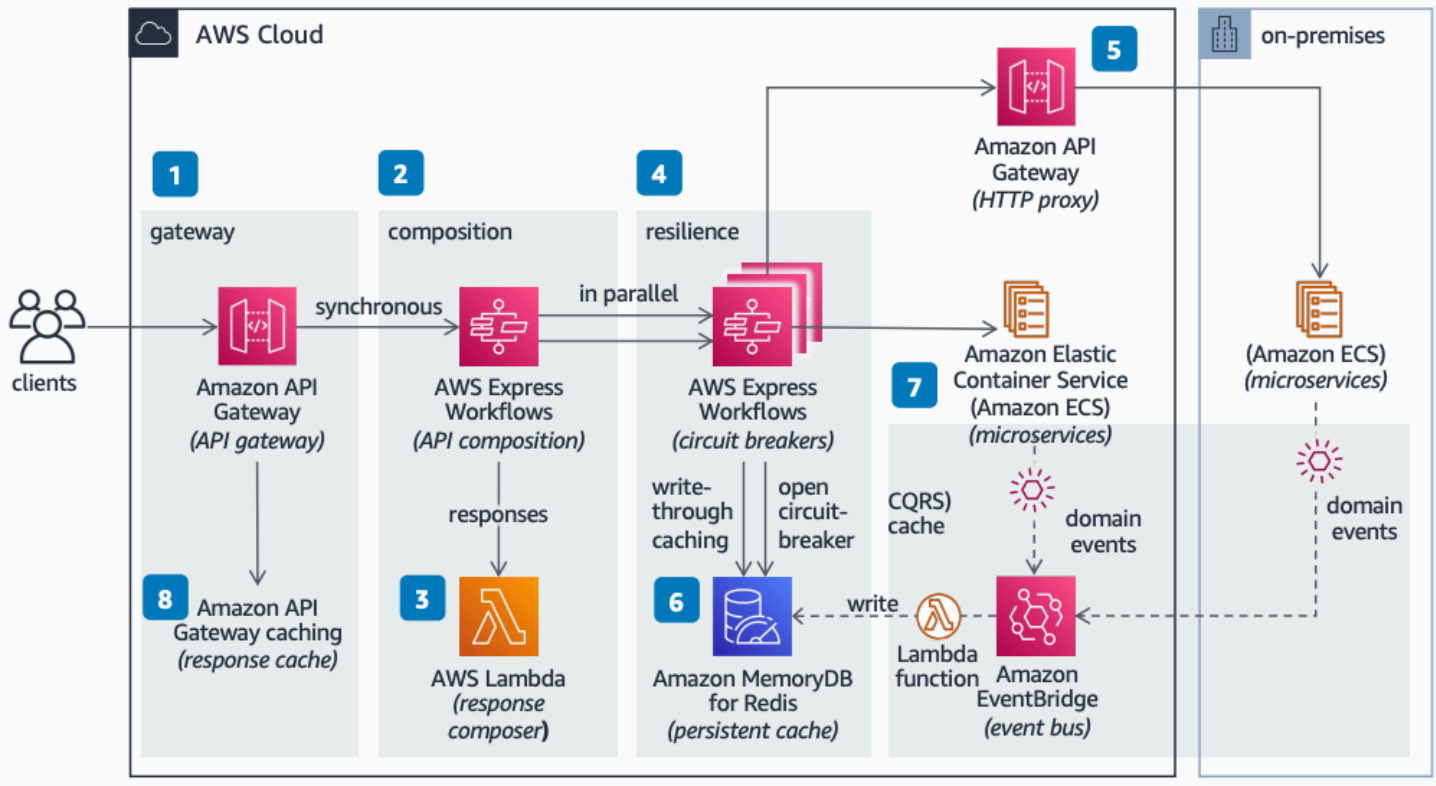
Parallele API-Zusammensetzung i
Parallele API-Zusammensetzung in AWS 1
Weitere Informationen 2
Verlauf des Diagramms 2
..... iv

Parallele API-Zusammensetzung in AWS

Veröffentlichungsdatum: 17. Mai 2022 () Geschichte des Diagramms

Diese Architektur zeigt, wie mehrere Downstream-API-Endpunkte parallel oder nacheinander aufgerufen werden, um eine einzige aggregierte Antwort zu verfassen. Sie erstellen mithilfe von AWS Step Functions Express-Workflows einen generischen, parametrisierten Circuit-Breaker-Workflow und verwenden die Command Query Responsibility Segregation (CQRS), um einen persistenten und letztlich konsistenten Cache aufrechtzuerhalten.

Parallele API-Zusammensetzung in AWS



Die folgenden Schritte beschreiben die Architektur:

1. Erstellen Sie mithilfe von Amazon API Gateway ein API-Gateway, damit Kunden synchrone Webanfragen an Ihre Microservices senden können.
2. Verwenden Sie Step Functions Express Workflows, um einen Workflow mit parallelen Schritten zu erstellen, die mehrere Microservices gleichzeitig aufrufen.
3. Verwenden Sie eine AWS Lambda Funktion, um die zahlreichen Antworten von Downstream-Microservices zu einer einzigen aggregierten Antwort für Kunden zusammenzustellen.

4. Behandeln Sie parallele Anfragen mit Schutzschaltern, die mit parametrisierten, verschachtelten Step Functions Express-Workflows ausgestattet sind, um die Ausfallsicherheit zu erhöhen.
5. Verwenden Sie API Gateway als Proxy für HTTP-Anfragen, wenn Sie lokale Microservices aufrufen.
6. Erstellen Sie eine Amazon MemoryDB for Redis-Datenbank, um Microservice-Antworten mithilfe des Write-Through-Caching-Musters zwischenzuspeichern. https://docs.aws.amazon.com/whitepapers/latest/database-caching-strategies-using-redis/caching-patterns.html Wenn ein Downstream-Microservice nicht mehr verfügbar ist oder seine Latenz einen Schwellenwert erreicht, schließt sich der Kreislauf und liest alle Antworten direkt aus dem Cache.
7. Ergänzen Sie zwischengespeicherte Daten mithilfe des CQRS-Musters durch Domänenereignisse von Downstream-Microservices. Erstellen Sie einen Event-Bus mit Amazon und behandeln Sie Ereignisse dann mit einer Lambda-Funktion EventBridge, um die Domain-Aggregate in MemoryDB zu speichern.
8. Aktivieren Sie den API Gateway-Antwort-Cache und passen Sie die TTL-Werte und Filter (Time-to-Live) je nach Anforderungstyp an, um die Leistung zu steigern.

Weitere Informationen

Weitere Informationen finden Sie in den folgenden Ressourcen:

- AWS Architektur-Ikonen
- AWS Zentrum für Architektur
- AWS Well-Architected

Geschichte des Diagramms

Abonnieren Sie den RSS-Feed, um über Aktualisierungen dieses Referenzarchitekturdiagramms informiert zu werden.

Änderung	Beschreibung	Datum
Erste Veröffentlichung	Das Referenzarchitekturdiagramm wurde zuerst veröffentlicht.	17. Mai 2022

> **Note**
>
> Um RSS-Updates zu abonnieren, muss ein RSS-Plugin für den von Ihnen verwendeten Browser aktiviert sein.

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.