



Panduan Developer

Amazon MemoryDB



Amazon MemoryDB: Panduan Developer

Copyright © 2024 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 menghina atau mendiskreditkan Amazon. Semua merek dagang lain yang tidak dimiliki oleh Amazon merupakan kekayaan masing-masing pemiliknya, yang mungkin atau mungkin tidak berafiliasi, terkait dengan, atau disponsori oleh Amazon.

Table of Contents

Apa itu MemoryDB	1
Fitur MemoryDB	1
Komponen inti MemoryDB	2
Klaster	3
Simpul	4
Serpihan	4
Grup parameter	5
Grup subnet	5
Daftar kontrol akses	5
Pengguna	6
Layanan terkait	6
Memilih Wilayah dan Availability Zone	6
Menempatkan simpul Anda	8
Wilayah & titik akhir yang didukung	9
Mengakses MemoryDB	12
Keamanan MemoryDB	13
Memulai dengan MemoryDB	14
Mengatur	14
Buat AWS akun Anda	15
Memberikan akses programatis	16
Siapkan izin Anda (hanya pengguna MemoryDB baru)	18
Mengunduh dan Mengkonfigurasi CLI AWS	19
Langkah 1: Buat cluster	20
Membuat cluster MemoryDB	20
Menyiapkan otentikasi	31
Langkah 2: Otorisasi akses ke cluster	32
Langkah 3: Connect ke cluster	34
Temukan titik akhir klaster Anda	34
Connect ke cluster MemoryDB (Linux)	34
Langkah 4: Menghapus cluster	36
Apa yang saya lakukan selanjutnya?	38
Mengelola simpul	40
Node dan pecahan MemoryDB	40
Jenis simpul yang didukung	42

Simpul terpesan	44
Ikhtisar node yang dicadangkan	44
Mengganti simpul	56
Mengelola kluster	58
Tingkatan data	59
Praktik terbaik	60
Batasan	60
Harga tingkatan data	61
Pemantauan	61
Menggunakan tingkatan data	61
Memulihkan data dari snapshot ke cluster dengan tiering data diaktifkan	63
Menyiapkan kluster	64
Menentukan kebutuhan Anda	64
Membuat kluster	68
Melihat detail kluster	69
Mengubah kluster	74
Menambahkan/Menghapus node dari cluster	77
Mengakses kluster Anda	79
Berikan akses ke cluster Anda	79
Mengakses MemoryDB dari luar AWS	81
Menemukan titik akhir koneksi	87
Serpihan	90
Menemukan nama pecahan	91
Mengelola implementasi MemoryDB Anda	95
Versi mesin	95
Redis OSS 7.0 (ditingkatkan)	95
Redis OSS 7.0 (ditingkatkan)	96
Redis OSS 6.2 (ditingkatkan)	97
Meningkatkan versi mesin	98
Mulai menggunakan JSON	100
Ikhtisar tipe data Redis OSS JSON	101
Perintah yang Didukung	113
Menandai sumber daya MemoryDB Anda	155
Memantau biaya dengan tag	160
Mengelola tag menggunakan AWS CLI	161
Mengelola tag menggunakan API MemoryDB	165

Mengelola pemeliharaan	167
Praktik terbaik	169
Perintah Redis OSS Terbatas	170
Ketangguhan	171
Praktik terbaik: Pub/Sub dan Multiplexing I/O yang Ditingkatkan	173
Praktik terbaik: Mengubah ukuran klaster secara online	173
Memahami replikasi MemoryDB	174
Konsistensi	175
Replikasi dalam sebuah cluster	175
Meminimalkan waktu henti dengan Multi-AZ	176
Mengubah jumlah replika	184
Melakukan snapshot dan pemulihan	194
Batasan	195
Biaya	195
Menjadwalkan snapshot otomatis	196
Membuat snapshot manual	197
Membuat snapshot akhir	200
Menggambarkan snapshot	202
Menyalin snapshot	205
Mengekspor snapshot	208
Memulihkan dari snapshot	218
Menyemai cluster dengan snapshot	224
Menandai snapshot	230
Menghapus snapshot	231
Penskalaan	232
Penskalaan cluster MemoryDB	234
Mengonfigurasi parameter mesin menggunakan grup parameter	256
Manajemen parameter	258
Tingkatan grup parameter	259
Membuat grup parameter	260
Membuat daftar grup parameter berdasarkan nama	264
Membuat daftar nilai grup parameter	269
Mengubah grup parameter	270
Menghapus grup parameter	273
Parameter spesifik Redis OSS	275
Tutorial: Mengonfigurasi fungsi Lambda untuk mengakses MemoryDB di VPC Amazon	293

Langkah 1: Buat cluster	293
Langkah 2: Buat fungsi Lambda	296
Langkah 3: Uji fungsi Lambda	300
Langkah 4: Bersihkan (Opsional)	301
Pencarian vektor	303
Ikhtisar pencarian vektor	303
Indeks dan ruang kunci	304
Jenis bidang indeks	305
Algoritma indeks vektor	306
Ekspresi kueri pencarian vektor	307
Perintah INFO	310
Keamanan pencarian vektor	312
Kasus penggunaan	313
Pengambilan Generasi Augmented () RAG	313
Cache Semantik Tahan Lama	313
Deteksi penipuan	314
Kasus penggunaan lainnya	315
Fitur pencarian vektor dan batas	315
Ketersediaan pencarian vektor	315
Pembatasan parametrik	316
Batas penskalaan	316
Pembatasan operasional	317
Impor/ekspor snapshot dan Migrasi Langsung	317
Konsumsi memori	317
Keluar dari Memori saat mengisi ulang	321
Transaksi	321
Menggunakan AWS Management Console	321
Menggunakan AWS Command Line Interface	322
Perintah pencarian vektor	322
KAKI. CREATE	323
KAKI. SEARCH	327
KAKI. AGGREGATE	330
KAKI. DROPINDEX	331
KAKI. INFO	331
KAKI. _ LIST	334
KAKI. ALIASADD	334

KAKI. ALIASDEL	335
KAKI. ALIASUPDATE	335
KAKI. _ ALIASLIST	335
KAKI. PROFILE	336
KAKI. EXPLAIN	336
KAKI. EXPLAINCLI	336
Keamanan	338
Perlindungan data	339
Keamanan data di MemoryDB	340
Enkripsi Diam	341
Enkripsi bergerak (TLS)	343
Mengautentikasi pengguna dengan ACL	344
Autentikasi dengan IAM	359
Pengelolaan identitas dan akses	367
Audiens	367
Mengautentikasi dengan identitas	368
Mengelola akses menggunakan kebijakan	372
Bagaimana MemoryDB bekerja dengan IAM	374
Contoh kebijakan berbasis identitas	384
Pemecahan Masalah	387
Kontrol akses	389
Gambaran umum pengelolaan akses	390
Pencatatan dan pemantauan	418
Pemantauan CloudWatch dengan	419
Pemantauan peristiwa	439
Mencatat panggilan API MemoryDB dengan AWS CloudTrail	452
Validasi kepatuhan	459
Keamanan infrastruktur	460
Privasi lalu lintas antar jaringan	460
MemoryDB dan Amazon VPC	461
Subnet dan grup subnet	473
MemoryDB API dan antarmuka titik akhir VPC ()AWS PrivateLink	487
Pembaruan layanan	490
Mengelola pembaruan layanan	490
Referensi	494
Menggunakan API MemoryDB	495

Menggunakan API kueri	495
Pustaka yang tersedia	498
Memecahkan masalah aplikasi	499
Kuota	501
Riwayat dokumen	502
.....	dv

Apa itu MemoryDB

MemoryDB adalah layanan database dalam memori yang tahan lama yang memberikan kinerja ultra-cepat. Ini dibuat khusus untuk aplikasi modern dengan arsitektur layanan mikro.

MemoryDB kompatibel dengan Redis OSS, penyimpanan data open source yang populer, memungkinkan Anda untuk dengan cepat membangun aplikasi menggunakan struktur data Redis OSS yang fleksibel dan ramah yang sama, API, dan perintah yang sudah mereka gunakan saat ini. Dengan MemoryDB, semua data Anda disimpan dalam memori, yang memungkinkan Anda mencapai baca mikrodetik dan latensi penulisan milidetik satu digit dan throughput tinggi. MemoryDB juga menyimpan data secara tahan lama di beberapa Availability Zones (AZ) menggunakan log transaksional multi-AZ untuk mengaktifkan failover cepat, pemulihan database, dan restart node.

Memberikan kinerja dalam memori dan daya tahan multi-AZ, MemoryDB dapat digunakan sebagai database utama berkinerja tinggi untuk aplikasi layanan mikro Anda, menghilangkan kebutuhan untuk mengelola cache dan database yang tahan lama secara terpisah.

Topik

- [Fitur MemoryDB](#)
- [Komponen inti MemoryDB](#)
- [Layanan terkait](#)
- [Memilih Wilayah dan Availability Zone](#)
- [Mengakses MemoryDB](#)
- [Keamanan MemoryDB](#)

Fitur MemoryDB

MemoryDB adalah layanan database dalam memori yang tahan lama yang memberikan kinerja ultra-cepat. Fitur MemoryDB meliputi:

- Konsistensi yang kuat untuk node primer dan konsistensi akhirnya dijamin untuk node replika. Untuk informasi selengkapnya, lihat [Konsistensi](#).
- Latensi baca mikrodetik dan latensi tulis milidetik satu digit dengan hingga 160 juta TPS per cluster.
- Struktur dan API data Redis OSS yang fleksibel dan ramah. Mudah membangun aplikasi baru atau memigrasikan aplikasi Redis OSS yang ada tanpa modifikasi.

- Daya tahan data menggunakan log transaksional multi-AZ yang menyediakan pemulihan database yang cepat dan restart.
- Ketersediaan multi-AZ dengan failover otomatis, dan deteksi dan pemulihan dari kegagalan node.
- Skala secara horizontal dengan mudah dengan menambahkan dan menghapus node atau secara vertikal dengan pindah ke tipe node yang lebih besar atau lebih kecil. Anda dapat menskalakan throughput tulis dengan menambahkan pecahan dan skala throughput baca dengan menambahkan replika.
- ead-after-write Konsistensi R untuk node primer dan konsistensi akhirnya dijamin untuk node replika.
- MemoryDB mendukung enkripsi dalam perjalanan, enkripsi saat istirahat dan otentikasi pengguna melalui. [Mengautentikasi pengguna dengan Daftar Kontrol Akses \(ACL\)](#)
- Snapshot otomatis di Amazon S3 dengan retensi hingga 35 hari.
- Support hingga 500 node dan penyimpanan lebih dari 100 TB per cluster (dengan 1 replika per pecahan).
- Enkripsi dalam perjalanan dengan TLS dan enkripsi saat istirahat dengan kunci. AWS KMS
- Otentikasi dan otorisasi pengguna dengan Redis OSS. [Mengautentikasi pengguna dengan Daftar Kontrol Akses \(ACL\)](#)
- Support untuk jenis AWS instans Graviton2.
- Integrasi dengan AWS layanan lain seperti CloudWatch, Amazon VPC, CloudTrail, dan Amazon SNS untuk pemantauan, keamanan, dan pemberitahuan.
- Penambalan dan peningkatan perangkat lunak yang dikelola sepenuhnya.
- AWS Integrasi Identity and Access Management (IAM) dan kontrol akses berbasis tag untuk API manajemen.

Komponen inti MemoryDB

Berikut ini, Anda dapat menemukan ikhtisar komponen utama dari penyebaran MemoryDB.

Topik

- [Klaster](#)
- [Simpul](#)
- [Serpihan](#)
- [Grup parameter](#)

- [Grup Subjaringan](#)
- [Daftar Kontrol Akses](#)
- [Pengguna](#)

Klaster

Cluster adalah kumpulan dari satu atau lebih node yang melayani satu dataset. Dataset MemoryDB dipartisi menjadi pecahan, dan setiap pecahan memiliki simpul utama dan hingga 5 node replika opsional. Node utama melayani permintaan baca dan tulis, sedangkan replika hanya melayani permintaan baca. Sebuah node primer dapat failover ke node replika, mempromosikan replika itu ke node primer baru untuk pecahan itu. MemoryDB menjalankan Redis OSS sebagai mesin database, dan ketika Anda membuat cluster, Anda menentukan versi Redis OSS untuk cluster Anda. Anda dapat membuat dan memodifikasi cluster menggunakan AWS CLI, MemoryDB API, atau file. AWS Management Console

Setiap cluster MemoryDB menjalankan versi mesin Redis OSS. Setiap versi mesin Redis OSS memiliki fitur yang didukung sendiri. Selain itu, setiap versi mesin Redis OSS memiliki seperangkat parameter dalam grup parameter yang mengontrol perilaku cluster yang dikelolanya.

Kapasitas komputasi dan memori cluster ditentukan oleh jenis simpulnya. Anda dapat memilih jenis instans yang paling sesuai untuk kebutuhan Anda. Jika kebutuhan Anda berubah seiring waktu, Anda dapat mengubah jenis simpul. Untuk informasi, lihat [Jenis simpul yang didukung](#).

Note

Untuk informasi harga tentang jenis node MemoryDB, lihat harga [MemoryDB](#).

Anda menjalankan klaster di cloud pribadi virtual (VPC) menggunakan layanan Amazon Virtual Private Cloud (Amazon VPC). Saat menggunakan VPC, Anda dapat mengontrol lingkungan jaringan virtual Anda. Anda dapat memilih rentang alamat IP Anda sendiri, membuat subnet, serta mengonfigurasi perutean dan daftar kontrol akses. MemoryDB mengelola snapshot, patching perangkat lunak, deteksi kegagalan otomatis, dan pemulihan. Tidak ada biaya tambahan untuk menjalankan klaster Anda di dalam VPC. Untuk informasi selengkapnya tentang penggunaan Amazon VPC dengan MemoryDB, lihat [MemoryDB dan Amazon VPC](#)

Banyak operasi MemoryDB ditargetkan pada cluster:

- Membuat klaster
- Mengubah klaster
- Mengambil snapshot dari sebuah cluster
- Menghapus klaster
- Melihat elemen di klaster
- Menambahkan atau menghapus tag alokasi biaya ke dan dari klaster

Untuk informasi selengkapnya, lihat topik terkait berikut:

- [Mengelola klaster](#) dan [Mengelola simpul](#)

Informasi tentang klaster, simpul, dan operasi terkait.

- [Ketahanan dalam MemoryDB](#)

Informasi tentang peningkatan toleransi kesalahan klaster Anda.

Simpul

Node adalah blok bangunan terkecil dari penerapan MemoryDB dan berjalan menggunakan instans Amazon EC2. Setiap node menjalankan versi Redis OSS yang dipilih saat Anda membuat cluster Anda. Sebuah simpul milik pecahan yang termasuk dalam sebuah cluster.

Setiap node menjalankan instance mesin pada versi yang dipilih saat Anda membuat cluster Anda. Jika perlu, Anda dapat menskalakan node dalam cluster naik atau turun ke jenis yang berbeda. Untuk informasi selengkapnya, lihat [Penskalaan](#).

Setiap node dalam cluster adalah tipe node yang sama. Beberapa jenis node didukung, masing-masing dengan jumlah memori yang bervariasi. Untuk daftar jenis simpul yang didukung, lihat [Jenis simpul yang didukung](#).

Untuk informasi selengkapnya tentang simpul, lihat [Mengelola simpul](#).

Serpihan

Shard adalah pengelompokan satu hingga 6 node, dengan satu berfungsi sebagai simpul tulis utama dan 5 lainnya berfungsi sebagai replika baca. Cluster MemoryDB selalu memiliki setidaknya satu pecahan.

Cluster MemoryDB dapat memiliki hingga 500 pecahan, dengan data Anda dipartisi di seluruh pecahan. Sebagai contoh, Anda dapat memilih untuk mengonfigurasi sebuah klaster dengan 500 simpul yang berkisar antara 83 serpihan (satu primer dan 5 replika per serpihan) dan 500 serpihan (satu primer dan tanpa replika). Pastikan alamat IP yang tersedia mencukupi untuk mengakomodasi peningkatan tersebut. Kesalahan umumnya termasuk subnet dalam grup subnet memiliki rentang CIDR yang terlalu kecil atau subnet dibagikan dan banyak digunakan oleh klaster lainnya.

Serpihan beberapa simpul mengimplementasikan replikasi dengan memiliki satu simpul primer baca/tulis dan 1–5 simpul replika. Untuk informasi selengkapnya, lihat [Memahami replikasi MemoryDB](#).

Untuk informasi selengkapnya tentang serpihan, lihat [Menggunakan serpihan](#).

Grup parameter

Grup parameter adalah cara mudah untuk mengelola pengaturan runtime untuk Redis OSS di cluster Anda. Parameter digunakan untuk mengontrol penggunaan memori, ukuran item, dan banyak lagi. Grup parameter MemoryDB adalah kumpulan bernama parameter khusus mesin yang dapat Anda terapkan ke cluster, dan semua node di cluster itu dikonfigurasi dengan cara yang persis sama.

Untuk informasi lebih rinci tentang grup parameter MemoryDB, lihat [Mengonfigurasi parameter mesin menggunakan grup parameter](#)

Grup Subjaringan

Grup subnet adalah kumpulan subnet (biasanya pribadi) yang dapat Anda tentukan untuk kluster yang berjalan di lingkungan Amazon Virtual Private Cloud (VPC).

Saat membuat klaster di VPC Amazon, Anda dapat menentukan grup subnet atau menggunakan grup default yang disediakan. MemoryDB menggunakan grup subnet itu untuk memilih subnet dan alamat IP dalam subnet itu untuk dikaitkan dengan node Anda.

Untuk informasi lebih rinci tentang grup subnet MemoryDB, lihat [Subnet dan grup subnet](#)

Daftar Kontrol Akses

Daftar kontrol akses adalah kumpulan dari satu atau lebih pengguna. String akses mengikuti [aturan Redis OSS ACL](#) untuk mengotorisasi akses pengguna ke perintah dan data Redis OSS.

Untuk informasi lebih rinci tentang Daftar Kontrol Akses MemoryDB, lihat [Mengautentikasi pengguna dengan Daftar Kontrol Akses \(ACL\)](#)

Pengguna

Seorang pengguna memiliki nama pengguna dan kata sandi, dan digunakan untuk mengakses data dan mengeluarkan perintah pada cluster MemoryDB Anda. Pengguna adalah anggota dari Access Control List (ACL), yang dapat Anda gunakan untuk menentukan izin bagi pengguna tersebut di cluster MemoryDB. Untuk informasi selengkapnya, lihat [Mengautentikasi pengguna dengan Daftar Kontrol Akses \(ACL\)](#)

Layanan terkait

[ElastiCache \(Redis OSS\)](#)

Ketika memutuskan apakah akan menggunakan MemoryDB atau ElastiCache (Redis OSS) mempertimbangkan perbandingan berikut:

- MemoryDB adalah database dalam memori yang tahan lama untuk beban kerja yang membutuhkan database primer yang sangat cepat. Anda harus mempertimbangkan untuk menggunakan MemoryDB jika beban kerja Anda memerlukan basis data durabel yang memberikan performa sangat cepat (latensi baca mikrodetik dan penulisan satu digit milidetik). MemoryDB mungkin juga cocok untuk kasus penggunaan Anda jika Anda ingin membangun aplikasi menggunakan struktur data Redis OSS dan API dengan database primer yang tahan lama. Terakhir, Anda harus mempertimbangkan untuk menggunakan MemoryDB untuk menyederhanakan arsitektur aplikasi Anda dan menurunkan biaya dengan mengganti penggunaan basis data dengan cache untuk durabilitas dan performa.
- ElastiCache (Redis OSS) adalah layanan yang biasa digunakan untuk cache data dari database lain dan penyimpanan data menggunakan Redis OSS. Anda harus mempertimbangkan ElastiCache (Redis OSS) untuk beban kerja caching di mana Anda ingin mempercepat akses data dengan database utama atau penyimpanan data yang ada (kinerja baca dan tulis mikrodetik). Anda juga harus mempertimbangkan ElastiCache (Redis OSS) untuk kasus penggunaan di mana Anda ingin menggunakan struktur data Redis OSS dan API untuk mengakses data yang disimpan dalam database utama atau penyimpanan data.

Memilih Wilayah dan Availability Zone

AWS Sumber daya komputasi awan ditempatkan di fasilitas pusat data yang sangat tersedia. Untuk memberikan skalabilitas dan keandalan tambahan, fasilitas pusat data ini ditempatkan di beberapa lokasi fisik yang berbeda. Lokasi ini dikategorikan berdasarkan wilayah dan Zona Ketersediaan.

AWS Daerah besar dan tersebar luas ke lokasi geografis yang terpisah. Availability Zone adalah lokasi berbeda dalam AWS Wilayah yang dirancang untuk diisolasi dari kegagalan di Availability Zone lainnya. Mereka menyediakan konektivitas jaringan latensi rendah yang murah ke Availability Zone lainnya di Wilayah yang sama AWS .

 Important

Setiap wilayah bersifat independen sepenuhnya. Aktivitas MemoryDB apa pun yang Anda mulai (misalnya, membuat cluster) hanya berjalan di wilayah default Anda saat ini.

Untuk membuat atau menggunakan klaster di wilayah tertentu, gunakan titik akhir layanan regional yang terkait. Untuk titik akhir layanan, lihat [Wilayah & titik akhir yang didukung](#).

Menempatkan simpul Anda

Setiap cluster yang memiliki setidaknya satu replika harus tersebar di seluruh AZ. Satu-satunya cara Anda dapat menemukan semuanya dalam satu AZ adalah dengan cluster yang terdiri dari pecahan simpul tunggal.

Dengan menemukan node di AZ yang berbeda, MemoryDB menghilangkan kemungkinan kegagalan, seperti pemadaman listrik, dalam satu AZ akan menyebabkan hilangnya ketersediaan.

- [Membuat cluster MemoryDB](#)
- [Memodifikasi cluster MemoryDB](#)

Wilayah & titik akhir yang didukung

MemoryDB tersedia di beberapa AWS Wilayah. Ini berarti Anda dapat meluncurkan cluster MemoryDB di lokasi yang memenuhi kebutuhan Anda. Misalnya, Anda dapat meluncurkan di AWS Wilayah terdekat dengan pelanggan Anda, atau meluncurkan di AWS Wilayah tertentu untuk memenuhi persyaratan hukum tertentu. Selain itu, karena MemoryDB memperluas ketersediaan ke AWS Wilayah baru, MemoryDB mendukung dua MAJOR.MINOR versi terbaru pada waktu itu untuk Wilayah baru. Untuk informasi selengkapnya tentang versi MemoryDB, lihat [Versi mesin Redis OSS](#)

Secara default, AWS SDK, MemoryDB API AWS CLI, dan konsol MemoryDB mengacu pada Wilayah AS-Timur (Virginia N.). Saat MemoryDB memperluas ketersediaan ke wilayah baru, titik akhir baru untuk wilayah ini juga tersedia untuk digunakan dalam permintaan HTTP, AWS SDK, AWS CLI dan konsol Anda.

Setiap Wilayah dirancang untuk terisolasi sepenuhnya dari Wilayah lainnya. Di dalam setiap wilayah terdapat beberapa Availability Zone (AZ). Dengan meluncurkan node Anda di AZ yang berbeda, Anda mencapai toleransi kesalahan sebesar mungkin. Untuk informasi selengkapnya tentang wilayah dan Availability Zone, [Memilih Wilayah dan Availability Zone](#) lihat di awal topik ini.

Wilayah di mana MemoryDB didukung

Nama Wilayah/Wilayah	Titik Akhir	Protokol	
Wilayah AS Timur (Ohio) us-east-2	memory-db.us-east-2.amazonaws.com	HTTPS	
Wilayah AS Timur (Virginia Utara) us-east-1	memory-db.us-east-1.amazonaws.com	HTTPS	
Wilayah AS Barat (California Utara) us-west-1	memory-db.us-west-1.amazonaws.com	HTTPS	

Nama Wilayah/Wilayah	Titik Akhir	Protokol	
Wilayah AS Barat (Oregon) us-west-2	memory-db.us-west-2.amazonaws.com	HTTPS	
Wilayah Kanada (Pusat) ca-central-1	memory-db.ca-central-1.amazonaws.com	HTTPS	
Wilayah Asia Pasifik (Hong Kong) ap-east-1	memory-db.ap-east-1.amazonaws.com	HTTPS	
Wilayah Asia Pasifik (Mumbai) ap-south-1	memory-db.ap-south-1.amazonaws.com	HTTPS	
Wilayah Asia Pasifik (Tokyo) ap-northeast-1	memory-db.ap-northeast-1.amazonaws.com	HTTPS	
Wilayah Asia Pasifik (Seoul) ap-northeast-2	memory-db.ap-northeast-2.amazonaws.com	HTTPS	
Wilayah Asia Pasifik (Singapura) ap-southeast-1	memory-db.ap-southeast-1.amazonaws.com	HTTPS	

Nama Wilayah/Wilayah	Titik Akhir	Protokol	
Wilayah Asia Pasifik (Sydney) ap-southeast-2	memory-db.ap-southeast-2.amazonaws.com	HTTPS	
Wilayah Eropa (Frankfurt) eu-central-1	memory-db.eu-central-1.amazonaws.com	HTTPS	
Wilayah Eropa (Irlandia) eu-west-1	memory-db.eu-west-1.amazonaws.com	HTTPS	
Wilayah Eropa (London) eu-west-2	memory-db.eu-west-2.amazonaws.com	HTTPS	
Wilayah Eropa (Paris) eu-west-3	memory-db.eu-west-3.amazonaws.com	HTTPS	
Wilayah Eropa (Stockholm) eu-north-1	memory-db.eu-north-1.amazonaws.com	HTTPS	
Wilayah Eropa (Milan) eu-south-1	memory-db.eu-south-1.amazonaws.com	HTTPS	
Wilayah Amerika Selatan (Sao Paulo) sa-east-1	memory-db.sa-east-1.amazonaws.com	HTTPS	

Nama Wilayah/Wilayah	Titik Akhir	Protokol
Wilayah Tiongkok (Beijing) cn-north-1	memory-db.cn-north-1.amazonaws.com.cn	HTTPS
Wilayah Tiongkok (Ningxia) cn-northwest-1	memory-db.cn-northwest-1.amazonaws.com.cn	HTTPS

Untuk tabel AWS produk dan layanan menurut wilayah, lihat [Produk dan layanan menurut Wilayah](#).

Untuk tabel Availability Zone yang didukung dalam Wilayah, lihat [Subnet dan grup subnet](#).

Mengakses MemoryDB

Setiap titik akhir cluster MemoryDB berisi alamat dan port. Endpoint cluster ini mendukung protokol Redis OSS Cluster untuk memungkinkan klien menemukan peran, alamat ip, dan slot tertentu untuk setiap node di cluster. Ketika node primer gagal dan replika dipromosikan sebagai gantinya, Anda dapat terhubung ke titik akhir cluster untuk menemukan primer baru menggunakan protokol Redis OSS Cluster.

Anda perlu terhubung ke titik akhir cluster untuk menemukan titik akhir node menggunakan cluster nodes atau cluster slots perintah. Setelah menemukan node yang tepat untuk kunci, Anda dapat terhubung langsung ke node untuk permintaan baca/tulis. Klien Redis OSS dapat menggunakan titik akhir cluster untuk secara otomatis terhubung ke node yang benar.

Untuk memecahkan masalah node tertentu dalam klaster, Anda juga dapat menggunakan titik akhir khusus node, tetapi ini tidak diperlukan untuk penggunaan normal.

Untuk menemukan titik akhir cluster, lihat berikut ini:

- [Menemukan Titik Akhir untuk Cluster MemoryDB \(CLI\)AWS](#)
- [Menemukan Endpoint untuk Cluster MemoryDB \(MemoryDB API\)](#)

Untuk menghubungkan ke node atau cluster, lihat [Menghubungkan ke node MemoryDB menggunakan redis-cli](#).

Keamanan MemoryDB

Keamanan untuk MemoryDB dikelola pada tiga tingkatan:

- Untuk mengontrol siapa yang dapat melakukan tindakan manajemen pada cluster dan node MemoryDB, Anda menggunakan AWS Identity and Access Management (IAM). Saat Anda terhubung AWS menggunakan kredensial IAM, AWS akun Anda harus memiliki kebijakan IAM yang memberikan izin yang diperlukan untuk melakukan operasi. Untuk informasi selengkapnya, lihat [Manajemen identitas dan akses di MemoryDB](#)
- Untuk mengontrol tingkat akses ke cluster, Anda membuat pengguna dengan izin tertentu dan menentukannya ke Daftar Kontrol Akses (ACL). ACL, pada gilirannya, kemudian dikaitkan dengan satu atau lebih cluster. Untuk informasi selengkapnya, lihat [Mengautentikasi pengguna dengan Daftar Kontrol Akses \(ACL\)](#).
- Cluster MemoryDB harus dibuat di cloud pribadi virtual (VPC) berdasarkan layanan Amazon VPC. Untuk mengontrol perangkat dan instans Amazon EC2 mana yang dapat membuka koneksi ke titik akhir dan port node untuk cluster MemoryDB di VPC, Anda menggunakan grup keamanan VPC. Anda dapat membuat koneksi titik akhir dan port ini menggunakan Keamanan Lapisan Pengangkutan (TLS)/Lapisan Soket Aman (SSL). Selain itu, aturan firewall di perusahaan Anda dapat mengontrol apakah perangkat yang berjalan di perusahaan Anda dapat membuka koneksi ke cluster MemoryDB. Untuk informasi selengkapnya tentang VPC, lihat [MemoryDB dan Amazon VPC](#).

Untuk informasi tentang konfigurasi keamanan, lihat [Keamanan di MemoryDB](#).

Memulai dengan MemoryDB

Latihan ini mengarahkan Anda melalui langkah-langkah untuk membuat, memberikan akses ke, terhubung ke, dan akhirnya menghapus kluster MemoryDB menggunakan MemoryDB Management Console.

Note

Untuk tujuan latihan ini, kami sarankan Anda menggunakan opsi Easy create saat membuat cluster dan kembali ke dua opsi lainnya setelah Anda menjelajahi lebih lanjut fitur MemoryDB.

Topik

- [Mengatur](#)
- [Langkah 1: Buat cluster](#)
- [Langkah 2: Otorisasi akses ke cluster](#)
- [Langkah 3: Connect ke cluster](#)
- [Langkah 4: Menghapus cluster](#)
- [Apa yang saya lakukan selanjutnya?](#)

Mengatur

Berikut ini, Anda dapat menemukan topik yang menggambarkan tindakan satu kali yang harus Anda ambil untuk mulai menggunakan MemoryDB.

Topik

- [Buat AWS akun Anda](#)
- [Memberikan akses programatis](#)
- [Siapkan izin Anda \(hanya pengguna MemoryDB baru\)](#)
- [Mengunduh dan Mengkonfigurasi CLI AWS](#)

Buat AWS akun Anda

Mendaftar untuk Akun AWS

Jika Anda tidak memiliki Akun AWS, selesaikan langkah-langkah berikut untuk membuatnya.

Untuk mendaftar untuk Akun AWS

1. Buka <https://portal.aws.amazon.com/billing/signup>.
2. Ikuti petunjuk online.

Bagian dari prosedur pendaftaran melibatkan tindakan menerima panggilan telepon dan memasukkan kode verifikasi di keypad telepon.

Saat Anda mendaftar untuk sebuah Akun AWS, sebuah Pengguna root akun AWS dibuat. Pengguna root memiliki akses ke semua AWS layanan dan sumber daya di akun. Sebagai praktik keamanan terbaik, tetapkan akses administratif ke pengguna, dan gunakan hanya pengguna root untuk melakukan [tugas yang memerlukan akses pengguna root](#).

AWS mengirim Anda email konfirmasi setelah proses pendaftaran selesai. Anda dapat melihat aktivitas akun Anda saat ini dan mengelola akun Anda dengan mengunjungi <https://aws.amazon.com/> dan memilih Akun Saya.

Buat pengguna dengan akses administratif

Setelah Anda mendaftar Akun AWS, amankan Pengguna root akun AWS, aktifkan AWS IAM Identity Center, dan buat pengguna administratif sehingga Anda tidak menggunakan pengguna root untuk tugas sehari-hari.

Amankan Anda Pengguna root akun AWS

1. Masuk ke [AWS Management Console](#) sebagai pemilik akun dengan memilih pengguna Root dan memasukkan alamat Akun AWS email Anda. Di laman berikutnya, masukkan kata sandi.

Untuk bantuan masuk dengan menggunakan pengguna root, lihat [Masuk sebagai pengguna root](#) di AWS Sign-In Panduan Pengguna.

2. Mengaktifkan autentikasi multi-faktor (MFA) untuk pengguna root Anda.

Untuk petunjuk, lihat [Mengaktifkan perangkat MFA virtual untuk pengguna Akun AWS root \(konsol\) Anda](#) di Panduan Pengguna IAM.

Buat pengguna dengan akses administratif

1. Aktifkan Pusat Identitas IAM.

Untuk mendapatkan petunjuk, silakan lihat [Mengaktifkan AWS IAM Identity Center](#) di Panduan Pengguna AWS IAM Identity Center .

2. Di Pusat Identitas IAM, berikan akses administratif ke pengguna.

Untuk tutorial tentang menggunakan Direktori Pusat Identitas IAM sebagai sumber identitas Anda, lihat [Mengkonfigurasi akses pengguna dengan default Direktori Pusat Identitas IAM](#) di Panduan AWS IAM Identity Center Pengguna.

Masuk sebagai pengguna dengan akses administratif

- Untuk masuk dengan pengguna Pusat Identitas IAM, gunakan URL masuk yang dikirim ke alamat email saat Anda membuat pengguna Pusat Identitas IAM.

Untuk bantuan masuk menggunakan pengguna Pusat Identitas IAM, lihat [Masuk ke portal AWS akses](#) di Panduan AWS Sign-In Pengguna.

Tetapkan akses ke pengguna tambahan

1. Di Pusat Identitas IAM, buat set izin yang mengikuti praktik terbaik menerapkan izin hak istimewa paling sedikit.

Untuk petunjuknya, lihat [Membuat set izin](#) di Panduan AWS IAM Identity Center Pengguna.

2. Tetapkan pengguna ke grup, lalu tetapkan akses masuk tunggal ke grup.

Untuk petunjuk, lihat [Menambahkan grup](#) di Panduan AWS IAM Identity Center Pengguna.

Memberikan akses programatis

Pengguna membutuhkan akses terprogram jika mereka ingin berinteraksi dengan AWS luar. AWS Management Console Cara untuk memberikan akses terprogram tergantung pada jenis pengguna yang mengakses AWS.

Untuk memberi pengguna akses programatis, pilih salah satu opsi berikut.

Pengguna mana yang membutuhkan akses programatis?	Untuk	Oleh
Identitas tenaga kerja (Pengguna yang dikelola di Pusat Identitas IAM)	Gunakan kredensial sementara untuk menandatangani permintaan terprogram ke AWS CLI, AWS SDK, atau API. AWS	Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan. <ul style="list-style-type: none"> • Untuk AWS CLI, lihat Mengkonfigurasi yang akan AWS CLI digunakan AWS IAM Identity Center dalam Panduan AWS Command Line Interface Pengguna. • Untuk AWS SDK, alat, dan AWS API, lihat otentikasi Pusat Identitas IAM di Panduan Referensi AWS SDK dan Alat.
IAM	Gunakan kredensial sementara untuk menandatangani permintaan terprogram ke AWS CLI, AWS SDK, atau API. AWS	Mengikuti petunjuk dalam Menggunakan kredensial sementara dengan AWS sumber daya di Panduan Pengguna IAM.
IAM	(Tidak direkomendasikan) Gunakan kredensial jangka panjang untuk menandatangani permintaan terprogram ke AWS CLI, AWS SDK, atau API. AWS	Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan. <ul style="list-style-type: none"> • Untuk mengetahui AWS CLI, lihat Mengotentikasi menggunakan kredensial pengguna IAM di Panduan Pengguna. AWS Command Line Interface

Pengguna mana yang membutuhkan akses programatis?	Untuk	Oleh
		<ul style="list-style-type: none"> • Untuk AWS SDK dan alat bantu, lihat Mengautentikasi menggunakan kredensial jangka panjang di Panduan Referensi AWS SDK dan Alat. • Untuk AWS API, lihat Mengelola kunci akses untuk pengguna IAM di Panduan Pengguna IAM.

Topik terkait:

- [Apa itu IAM?](#) dalam Panduan Pengguna IAM.
- [AWS Kredensi Keamanan dalam Referensi AWS Umum](#).

Siapkan izin Anda (hanya pengguna MemoryDB baru)

Untuk memberikan akses, menambahkan izin ke pengguna, grup, atau peran Anda:

- Pengguna dan grup di AWS IAM Identity Center:

Buat rangkaian izin. Ikuti instruksi di [Buat rangkaian izin](#) di Panduan Pengguna AWS IAM Identity Center .

- Pengguna yang dikelola di IAM melalui penyedia identitas:

Buat peran untuk federasi identitas. Ikuti instruksi dalam [Membuat peran untuk penyedia identitas pihak ketiga \(federasi\)](#) di Panduan Pengguna IAM.

- Pengguna IAM:

- Buat peran yang dapat diambil pengguna Anda. Ikuti instruksi dalam [Membuat peran untuk pengguna IAM](#) dalam Panduan Pengguna IAM.

- (Tidak disarankan) Pasang kebijakan langsung ke pengguna atau tambahkan pengguna ke grup pengguna. Ikuti instruksi dalam [Menambahkan izin ke pengguna \(konsol\)](#) dalam Panduan Pengguna IAM.

MemoryDB membuat dan menggunakan peran terkait layanan untuk menyediakan sumber daya dan mengakses sumber AWS daya dan layanan lain atas nama Anda. Agar MemoryDB membuat peran terkait layanan untuk Anda, gunakan kebijakan -managed bernama.

AWSAmazonMemoryDBFullAccess Peran ini dilengkapi sebelumnya dengan izin yang diperlukan layanan untuk membuat peran terkait layanan untuk Anda.

Anda mungkin memutuskan untuk tidak menggunakan kebijakan default dan sebagai gantinya menggunakan kebijakan yang dikelola kustom. Dalam hal ini, pastikan bahwa Anda memiliki izin untuk menelepon `iam:createServiceLinkedRole` atau bahwa Anda telah membuat peran terkait layanan MemoryDB.

Untuk informasi lain, lihat yang berikut ini:

- [Membuat Kebijakan Baru \(IAM\)](#)
- [AWS kebijakan -managed \(telah ditentukan\) untuk MemoryDB](#)
- [Menggunakan Peran Tertaut Layanan untuk MemoryDB](#)

Mengunduh dan Mengkonfigurasi CLI AWS

AWS CLI Tersedia di <http://aws.amazon.com/cli>. CLI berjalan di Windows, MacOS, atau Linux. Setelah Anda mengunduh AWS CLI, ikuti langkah-langkah berikut untuk menginstal dan mengkonfigurasinya:

1. Lihat ke [Panduan Pengguna Antar Muka Baris Perintah AWS](#).
2. Ikuti instruksi untuk [Menginstal AWS CLI dan Mengkonfigurasi CLI. AWS](#)

Langkah 1: Buat cluster

Sebelum membuat klaster untuk tujuan produksi, Anda tentu perlu mempertimbangkan pengaturan konfigurasi klaster untuk memenuhi kebutuhan bisnis Anda. Masalah terkait itu dibahas di bagian [Menyiapkan klaster](#). Untuk tujuan latihan Memulai ini, Anda dapat menerima nilai konfigurasi default yang diterapkan.

Klaster yang Anda buat akan berjalan langsung, dan tidak berjalan di sandbox. Anda akan dikenakan biaya penggunaan MemoryDB standar untuk instance sampai Anda menghapusnya. Jumlah biayanya cukup kecil (biasanya kurang dari satu dolar) jika Anda menyelesaikan latihan yang dijelaskan di sini dalam satu sesi dan menghapus klaster itu ketika Anda sudah selesai. [Untuk informasi selengkapnya tentang tingkat penggunaan MemoryDB, lihat MemoryDB.](#)

Klaster Anda diluncurkan dalam cloud privat virtual (VPC) berdasarkan layanan Amazon VPC.

Membuat cluster MemoryDB

Contoh berikut menunjukkan cara membuat cluster menggunakan AWS Management Console, AWS CLI dan MemoryDB API.

Membuat cluster (Konsol)

Untuk membuat cluster menggunakan konsol MemoryDB

1. [Masuk ke AWS Management Console dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/.](https://console.aws.amazon.com/memorydb/)
2. Pilih Cluster Di panel navigasi kiri dan kemudian pilih Buat.

Easy create

1. Lengkapi bagian Konfigurasi. Ini mengonfigurasi tipe node dan konfigurasi default cluster Anda. Pilih ukuran memori dan kinerja jaringan yang sesuai yang Anda butuhkan dari opsi berikut:
 - Produksi
 - Dev/Uji
 - Demo
2. Lengkapi bagian Info Cluster.

- a. Pada Nama, masukkan nama untuk klaster Anda.

Batasan penamaan klaster adalah sebagai berikut:

- Harus berisi 1–40 karakter alfanumerik atau tanda hubung.
- Harus diawali dengan huruf.
- Tidak boleh berisi dua tanda hubung berurutan.
- Tidak boleh diakhiri dengan tanda hubung.

- b. Pada kotak Deskripsi, masukkan deskripsi untuk klaster ini.

3. Lengkapi bagian grup Subnet:

- Untuk grup Subnet, buat grup subnet baru atau pilih yang sudah ada dari daftar yang tersedia yang ingin Anda terapkan ke cluster ini. Jika Anda membuat yang baru:
 - Masukkan Nama
 - Masukkan Deskripsi
 - Jika Anda mengaktifkan Multi-AZ, grup subnet harus berisi setidaknya dua subnet yang berada dalam availability zone yang berbeda. Untuk informasi selengkapnya, lihat [Subnet dan grup subnet](#).
 - Jika Anda membuat grup subnet baru dan tidak memiliki VPC yang ada, Anda akan diminta untuk membuat VPC. Untuk informasi lebih lanjut, lihat [Apa itu Amazon VPC?](#) di Panduan Pengguna Amazon VPC.

4. Untuk pencarian Vektor, Anda dapat Aktifkan kemampuan pencarian Vektor untuk menyimpan embeddings vektor dan melakukan pencarian vektor. Perhatikan bahwa ini akan memperbaiki nilai untuk kompatibilitas versi Redis OSS, grup Parameter dan Pecahan. Untuk informasi selengkapnya, lihat [Pencarian vektor](#).

5. Lihat pengaturan default:

Saat menggunakan Easy create, pengaturan cluster yang tersisa diatur secara default. Perhatikan bahwa beberapa pengaturan ini dapat diubah setelah pembuatan, seperti yang ditunjukkan oleh Editable setelah pembuatan.

6. Untuk Tag, Anda dapat menerapkan tag secara opsional untuk mencari dan memfilter cluster Anda atau melacak biaya Anda AWS .
7. Tinjau semua entri dan pilihan Anda, lalu lakukan koreksi yang diperlukan. Saat Anda siap, pilih Buat untuk meluncurkan klaster Anda, atau Batal untuk membatalkan operasi.

Setelah status klaster Anda menjadi tersedia, Anda dapat memberi EC2 akses ke klaster tersebut, terhubung dengannya, dan mulai menggunakannya. Untuk informasi selengkapnya, lihat [Langkah 2: Otorisasi akses ke cluster](#)

⚠ Important

Setelah klaster Anda tersedia, Anda akan ditagih untuk setiap jam atau jam parsial saat klaster aktif, meskipun Anda tidak sedang aktif menggunakannya. Untuk menghentikan tagihan untuk klaster ini, Anda harus menghapusnya. Lihat [Langkah 4: Menghapus cluster](#).

Create new cluster

1. Lengkapi bagian Info Cluster.

- a. Pada Nama, masukkan nama untuk klaster Anda.

Batasan penamaan klaster adalah sebagai berikut:

- Harus berisi 1–40 karakter alfanumerik atau tanda hubung.
- Harus diawali dengan huruf.
- Tidak boleh berisi dua tanda hubung berurutan.
- Tidak boleh diakhiri dengan tanda hubung.

- b. Pada kotak Deskripsi, masukkan deskripsi untuk klaster ini.

2. Lengkapi bagian grup Subnet:

- Untuk grup Subnet, buat grup subnet baru atau pilih yang sudah ada dari daftar yang tersedia yang ingin Anda terapkan ke cluster ini. Jika Anda membuat yang baru:
 - Masukkan Nama
 - Masukkan Deskripsi
 - Jika Anda mengaktifkan Multi-AZ, grup subnet harus berisi setidaknya dua subnet yang berada dalam availability zone yang berbeda. Untuk informasi selengkapnya, lihat [Subnet dan grup subnet](#).

- Jika Anda membuat grup subnet baru dan tidak memiliki VPC yang ada, Anda akan diminta untuk membuat VPC. Untuk informasi lebih lanjut, lihat [Apa itu Amazon VPC?](#) di Panduan Pengguna Amazon VPC.

3. Lengkapi bagian Pengaturan cluster:

- a. Untuk mengaktifkan kemampuan pencarian Vektor, Anda dapat mengaktifkan ini untuk menyimpan embeddings vektor dan melakukan pencarian vektor. Perhatikan bahwa ini akan memperbaiki nilai untuk kompatibilitas versi Redis OSS, grup Parameter dan Pecahan. Untuk informasi selengkapnya, lihat [Pencarian vektor](#).
- b. Untuk kompatibilitas versi Redis OSS, terima default. 6 . 2
- c. Untuk Port, terima port Redis OSS default 6379 atau, jika Anda memiliki alasan untuk menggunakan port yang berbeda, masukkan nomor port..
- d. Untuk grup Parameter, jika Anda telah mengaktifkan pencarian vektor, gunakan default `memorydb-redis7.search.preview`. Jika tidak, terima grup default `memorydb-redis7` parameter.

Grup parameter mengontrol parameter runtime dari klaster Anda. Untuk mengetahui informasi selengkapnya tentang grup parameter, lihat [Parameter spesifik Redis OSS](#).

- e. Untuk tipe Node, pilih nilai untuk tipe node (bersama dengan ukuran memori terkait) yang Anda inginkan.

Jika Anda memilih tipe node dari keluarga `r6gd`, Anda akan secara otomatis mengaktifkan data-tiering, yang membagi penyimpanan data antara memori dan SSD. Untuk informasi selengkapnya, lihat [Tingkatan data](#).

- f. Untuk Jumlah pecahan, pilih jumlah pecahan yang Anda inginkan untuk cluster ini. Untuk ketersediaan cluster Anda yang lebih tinggi, kami sarankan Anda menambahkan setidaknya 2 pecahan.

Anda dapat mengubah jumlah pecahan di cluster Anda secara dinamis. Untuk informasi selengkapnya, lihat [Penskalaan cluster MemoryDB](#).


- g. Untuk Replika per serpihan, pilih jumlah simpul replika baca yang Anda inginkan dalam setiap serpihan.

Pembatasan berikut ada:

- Jika Multi-AZ diaktifkan, pastikan bahwa Anda memiliki setidaknya satu replika per serpihan.
 - Replika akan berjumlah sama untuk setiap serpihan saat membuat klaster menggunakan konsol.
- h. Pilih Berikutnya
- i. Lengkapi bagian Pengaturan lanjutan:
- i. Untuk Grup keamanan, pilih grup keamanan yang Anda inginkan untuk klaster ini. Grup keamanan bertindak sebagai firewall untuk mengontrol akses jaringan ke klaster Anda. Anda dapat menggunakan grup keamanan default untuk VPC Anda atau membuat yang baru.

Untuk informasi selengkapnya tentang grup keamanan, lihat [Grup Keamanan untuk VPC Anda](#) dalam Panduan Pengguna Amazon VPC.

- ii. Untuk mengenkripsi data Anda, Anda memiliki opsi berikut:
- Enkripsi diam – Mengaktifkan enkripsi pada data yang disimpan di disk. Untuk informasi selengkapnya, lihat [Enkripsi Diam](#).

 Note

Anda memiliki opsi untuk menyediakan kunci enkripsi selain default dengan memilih kunci KMS AWS milik Pelanggan yang Dikelola dan memilih kunci.


- Enkripsi bergerak – Mengaktifkan enkripsi data selama pengiriman. Jika Anda memilih tidak ada enkripsi, maka daftar kontrol Akses terbuka yang disebut “akses terbuka” akan dibuat dengan pengguna default. Untuk informasi selengkapnya, lihat [Mengautentikasi pengguna dengan Daftar Kontrol Akses \(ACL\)](#).
- iii. Untuk Snapshot, secara opsional tentukan periode retensi snapshot dan jendela snapshot. Secara default, Aktifkan snapshot otomatis telah dipilih sebelumnya.
- iv. Untuk jendela Pemeliharaan opsional menentukan jendela pemeliharaan. Jendela pemeliharaan adalah waktu, umumnya satu jam panjangnya, setiap minggu ketika MemoryDB menjadwalkan pemeliharaan sistem untuk cluster Anda. Anda dapat mengizinkan MemoryDB untuk memilih hari dan waktu untuk jendela pemeliharaan Anda (Tidak ada preferensi), atau Anda dapat memilih hari, waktu, dan durasi

sendiri (Tentukan jendela pemeliharaan). Jika Anda memilih Tentukan periode pemeliharaan dari daftar, pilih Hari mulai, Waktu mulai, dan Durasi (dalam jam) untuk periode pemeliharaan. Semua waktu menggunakan zona waktu UTC.

Untuk informasi selengkapnya, lihat [Mengelola pemeliharaan](#).

- v. Untuk Pemberitahuan, pilih topik Amazon Simple Notification Service (Amazon SNS) yang sudah ada, atau pilih Input ARN Manual dan masukkan Nama Sumber Daya Amazon (ARN) topik. Amazon SNS memungkinkan Anda untuk mendorong pemberitahuan ke perangkat pintar yang terhubung ke Internet. Defaultnya adalah menonaktifkan notifikasi. Untuk informasi lebih lanjut, lihat <https://aws.amazon.com/sns/>.
- vi. Untuk Tag, Anda dapat menerapkan tag secara opsional untuk mencari dan memfilter cluster Anda atau melacak biaya Anda AWS .
- j. Tinjau semua entri dan pilihan Anda, lalu lakukan koreksi yang diperlukan. Saat Anda siap, pilih Buat untuk meluncurkan klaster Anda, atau Batal untuk membatalkan operasi.

Setelah status klaster Anda menjadi tersedia, Anda dapat memberi EC2 akses ke klaster tersebut, terhubung dengannya, dan mulai menggunakannya. Untuk informasi selengkapnya, lihat [Langkah 2: Otorisasi akses ke cluster](#)

 Important

Setelah klaster Anda tersedia, Anda akan ditagih untuk setiap jam atau jam parsial saat klaster aktif, meskipun Anda tidak sedang aktif menggunakannya. Untuk menghentikan tagihan untuk klaster ini, Anda harus menghapusnya. Lihat [Langkah 4: Menghapus cluster](#).

Restore from snapshots

Di bawah Sumber snapshot, pilih snapshot sumber untuk memigrasikan data. Untuk informasi selengkapnya, lihat [Melakukan snapshot dan pemulihan](#) .

Note

Jika Anda ingin cluster baru Anda mengaktifkan pencarian vektor, snapshot sumber juga harus mengaktifkan pencarian vektor.

Cluster target default ke pengaturan cluster sumber. Secara opsional, Anda dapat mengubah pengaturan berikut pada cluster target:

1. Info klaster

- a. Pada Nama, masukkan nama untuk klaster Anda.

Batasan penamaan klaster adalah sebagai berikut:

- Harus berisi 1–40 karakter alfanumerik atau tanda hubung.
- Harus diawali dengan huruf.
- Tidak boleh berisi dua tanda hubung berurutan.
- Tidak boleh diakhiri dengan tanda hubung.

- b. Pada kotak Deskripsi, masukkan deskripsi untuk klaster ini.

2. Grup subnet

- Untuk grup Subnet, buat grup subnet baru atau pilih yang sudah ada dari daftar yang tersedia yang ingin Anda terapkan ke cluster ini. Jika Anda membuat yang baru:
 - Masukkan Nama
 - Masukkan Deskripsi
 - Jika Anda mengaktifkan Multi-AZ, grup subnet harus berisi setidaknya dua subnet yang berada dalam availability zone yang berbeda. Untuk informasi selengkapnya, lihat [Subnet dan grup subnet](#).
 - Jika Anda membuat grup subnet baru dan tidak memiliki VPC yang ada, Anda akan diminta untuk membuat VPC. Untuk informasi lebih lanjut, lihat [Apa itu Amazon VPC?](#) di Panduan Pengguna Amazon VPC.

3. Pengaturan cluster

- a. Untuk mengaktifkan kemampuan pencarian Vektor, Anda dapat mengaktifkan ini untuk menyimpan embeddings vektor dan melakukan pencarian vektor. Perhatikan bahwa

ini akan memperbaiki nilai untuk kompatibilitas versi Redis OSS, grup Parameter dan Pecahan. Untuk informasi selengkapnya, lihat [Pencarian vektor](#).

- b. Untuk kompatibilitas versi Redis OSS, terima default. 6.2
- c. Untuk Port, terima port Redis OSS default 6379 atau, jika Anda memiliki alasan untuk menggunakan port yang berbeda, masukkan nomor port..
- d. Untuk grup Parameter, jika Anda telah mengaktifkan pencarian vektor, gunakan `default.memorydb-redis7.search.preview`. Jika tidak, terima grup `default.memorydb-redis7` parameter.

Grup parameter mengontrol parameter runtime dari klaster Anda. Untuk mengetahui informasi selengkapnya tentang grup parameter, lihat [Parameter spesifik Redis OSS](#).

- e. Untuk tipe Node, pilih nilai untuk tipe node (bersama dengan ukuran memori terkait) yang Anda inginkan.

Jika Anda memilih tipe node dari keluarga `r6gd`, Anda akan secara otomatis mengaktifkan `data-tiering`, yang membagi penyimpanan data antara memori dan SSD. Untuk informasi selengkapnya, lihat [Tingkatan data](#).

- f. Untuk Jumlah pecahan, pilih jumlah pecahan yang Anda inginkan untuk cluster ini. Untuk ketersediaan cluster Anda yang lebih tinggi, kami sarankan Anda menambahkan setidaknya 2 pecahan.

Anda dapat mengubah jumlah pecahan di cluster Anda secara dinamis. Untuk informasi selengkapnya, lihat [Penskalaan cluster MemoryDB](#).

- g. Untuk Replika per serpihan, pilih jumlah simpul replika baca yang Anda inginkan dalam setiap serpihan.


Pembatasan berikut ada:

- Jika Multi-AZ diaktifkan, pastikan bahwa Anda memiliki setidaknya satu replika per serpihan.
 - Replika akan berjumlah sama untuk setiap serpihan saat membuat klaster menggunakan konsol.
- h. Pilih Berikutnya
 - i. Pengaturan lanjutan

- i. Untuk Grup keamanan, pilih grup keamanan yang Anda inginkan untuk kluster ini. Grup keamanan bertindak sebagai firewall untuk mengontrol akses jaringan ke kluster Anda. Anda dapat menggunakan grup keamanan default untuk VPC Anda atau membuat yang baru.

Untuk informasi selengkapnya tentang grup keamanan, lihat [Grup Keamanan untuk VPC Anda](#) dalam Panduan Pengguna Amazon VPC.

- ii. Untuk mengenkripsi data Anda, Anda memiliki opsi berikut:
 - Enkripsi diam – Mengaktifkan enkripsi pada data yang disimpan di disk. Untuk informasi selengkapnya, lihat [Enkripsi Diam](#).

 Note

Anda memiliki opsi untuk menyediakan kunci enkripsi selain default dengan memilih kunci KMS AWS milik Pelanggan yang Dikelola dan memilih kunci.


- Enkripsi bergerak – Mengaktifkan enkripsi data selama pengiriman. Jika Anda memilih tidak ada enkripsi, maka daftar kontrol Akses terbuka yang disebut “akses terbuka” akan dibuat dengan pengguna default. Untuk informasi selengkapnya, lihat [Mengautentikasi pengguna dengan Daftar Kontrol Akses \(ACL\)](#).
- iii. Untuk Snapshot, secara opsional tentukan periode retensi snapshot dan jendela snapshot. Secara default, Aktifkan snapshot otomatis telah dipilih sebelumnya.
 - iv. Untuk jendela Pemeliharaan opsional menentukan jendela pemeliharaan. Jendela pemeliharaan adalah waktu, umumnya satu jam panjangnya, setiap minggu ketika MemoryDB menjadwalkan pemeliharaan sistem untuk cluster Anda. Anda dapat mengizinkan MemoryDB untuk memilih hari dan waktu untuk jendela pemeliharaan Anda (Tidak ada preferensi), atau Anda dapat memilih hari, waktu, dan durasi sendiri (Tentukan jendela pemeliharaan). Jika Anda memilih Tentukan periode pemeliharaan dari daftar, pilih Hari mulai, Waktu mulai, dan Durasi (dalam jam) untuk periode pemeliharaan. Semua waktu menggunakan zona waktu UTC.

Untuk informasi selengkapnya, lihat [Mengelola pemeliharaan](#).

- v. Untuk Pemberitahuan, pilih topik Amazon Simple Notification Service (Amazon SNS) yang sudah ada, atau pilih Input ARN Manual dan masukkan Nama Sumber

- Daya Amazon (ARN) topik. Amazon SNS memungkinkan Anda untuk mendorong pemberitahuan ke perangkat pintar yang terhubung ke Internet. Defaultnya adalah menonaktifkan notifikasi. Untuk informasi lebih lanjut, lihat <https://aws.amazon.com/sns/>.
- vi. Untuk Tag, Anda dapat menerapkan tag secara opsional untuk mencari dan memfilter cluster Anda atau melacak biaya Anda AWS .
 - j. Tinjau semua entri dan pilihan Anda, lalu lakukan koreksi yang diperlukan. Saat Anda siap, pilih Buat untuk meluncurkan klaster Anda, atau Batal untuk membatalkan operasi.

Setelah status klaster Anda menjadi tersedia, Anda dapat memberi EC2 akses ke klaster tersebut, terhubung dengannya, dan mulai menggunakannya. Untuk informasi selengkapnya, lihat [Langkah 2: Otorisasi akses ke cluster](#)

 Important

Setelah klaster Anda tersedia, Anda akan ditagih untuk setiap jam atau jam parsial saat klaster aktif, meskipun Anda tidak sedang aktif menggunakannya. Untuk menghentikan tagihan untuk klaster ini, Anda harus menghapusnya. Lihat [Langkah 4: Menghapus cluster](#).

Membuat klaster AWS (CLI)

Untuk membuat cluster menggunakan AWS CLI, lihat [create-cluster](#). Berikut adalah contohnya:

Untuk Linux, macOS, atau Unix:

```
aws memorydb create-cluster \  
  --cluster-name my-cluster \  
  --node-type db.r6g.large \  
  --acl-name my-acl \  
  --subnet-group my-sg
```

Untuk Windows:

```
aws memorydb create-cluster ^  
  --cluster-name my-cluster ^  
  --node-type db.r6g.large ^  
  --acl-name my-acl ^  
  --subnet-group my-sg
```

Anda harus mendapatkan respons JSON berikut:

```
{  
  "Cluster": {  
    "Name": "my-cluster",  
    "Status": "creating",  
    "NumberOfShards": 1,  
    "AvailabilityMode": "MultiAZ",  
    "ClusterEndpoint": {  
      "Port": 6379  
    },  
    "NodeType": "db.r6g.large",  
    "EngineVersion": "6.2",  
    "EnginePatchVersion": "6.2.6",  
    "ParameterGroupName": "default.memorydb-redis6",  
    "ParameterGroupStatus": "in-sync",  
    "SubnetGroupName": "my-sg",  
    "TLSEnabled": true,  
    "ARN": "arn:aws:memorydb:us-east-1:xxxxxxxxxxxx:cluster/my-cluster",  
    "SnapshotRetentionLimit": 0,  
    "MaintenanceWindow": "wed:03:00-wed:04:00",  
    "SnapshotWindow": "04:30-05:30",  
  }  
}
```

```
    "ACLName": "my-acl",
    "DataTiering": "false",
    "AutoMinorVersionUpgrade": true
  }
}
```

Anda dapat mulai menggunakan cluster setelah statusnya berubah menjadi `available`.

Important

Setelah klaster Anda tersedia, Anda akan ditagih untuk setiap jam atau jam parsial saat klaster aktif, meskipun Anda tidak sedang aktif menggunakannya. Untuk menghentikan tagihan untuk klaster ini, Anda harus menghapusnya. Lihat [Langkah 4: Menghapus cluster](#).

Membuat cluster (MemoryDB API)

Untuk membuat cluster menggunakan API MemoryDB, gunakan tindakan. [CreateCluster](#)

Important

Setelah klaster Anda tersedia, Anda akan ditagih untuk setiap jam atau jam parsial saat klaster aktif, meskipun Anda tidak menggunakannya. Untuk menghentikan tagihan untuk klaster ini, Anda harus menghapusnya. Lihat [Langkah 4: Menghapus cluster](#).

Menyiapkan otentikasi

Untuk informasi tentang menyiapkan otentikasi untuk klaster Anda, lihat [Autentikasi dengan IAM](#) dan [Mengautentikasi pengguna dengan Daftar Kontrol Akses \(ACL\)](#).

Langkah 2: Otorisasi akses ke cluster

Bagian ini mengasumsikan bahwa Anda telah memahami peluncuran dan koneksi ke instans Amazon EC2. Untuk informasi selengkapnya, lihat [Panduan Memulai Amazon EC2](#).

Cluster MemoryDB dirancang untuk diakses dari instans Amazon EC2. Mereka juga dapat diakses oleh aplikasi kontainer atau tanpa server yang berjalan di Amazon Elastic Container Service atau AWS Lambda Skenario yang paling umum adalah mengakses cluster MemoryDB dari instans Amazon EC2 di Amazon Virtual Private Cloud (Amazon VPC) yang sama, yang akan menjadi kasus untuk latihan ini.

Sebelum Anda dapat terhubung ke klaster dari instans EC2, Anda harus memberikan otorisasi pada instans EC2 untuk mengakses klaster tersebut.

Kasus penggunaan yang paling umum adalah saat aplikasi yang di-deploy pada instans EC2 perlu terhubung ke klaster di VPC yang sama. Cara paling sederhana untuk mengelola akses antara instans EC2 dan klaster di VPC yang sama adalah dengan melakukan tindakan berikut:

1. Buat grup keamanan VPC untuk klaster Anda. Grup keamanan ini dapat digunakan untuk membatasi akses ke cluster. Sebagai contoh, Anda dapat membuat aturan kustom untuk grup keamanan ini yang mengizinkan akses TCP menggunakan port yang Anda tetapkan untuk klaster saat Anda membuatnya dan alamat IP yang Anda gunakan untuk mengakses klaster tersebut.

Port default untuk cluster MemoryDB adalah 6379

2. Buat grup keamanan VPC untuk instans EC2 Anda (server web dan aplikasi). Jika diperlukan, grup keamanan ini dapat mengizinkan akses ke instans EC2 dari Internet menggunakan tabel perutean VPC. Sebagai contoh, Anda dapat menetapkan aturan pada grup keamanan ini untuk mengizinkan akses TCP ke instans EC2 melalui port 22.
3. Buat aturan khusus di grup keamanan untuk klaster Anda yang memungkinkan koneksi dari grup keamanan yang Anda buat untuk instans EC2 Anda. Hal ini akan mengizinkan semua anggota grup keamanan untuk mengakses klaster.

Untuk membuat aturan dalam grup keamanan VPC yang memungkinkan koneksi dari grup keamanan lain

1. [Masuk ke Konsol AWS Manajemen dan buka konsol VPC Amazon di https://console.aws.amazon.com/vpc.](https://console.aws.amazon.com/vpc)

2. Di panel navigasi kiri, pilih Grup Keamanan.
3. Pilih atau buat grup keamanan yang akan Anda gunakan untuk cluster Anda. Pada Aturan Masuk, pilih Edit Aturan Masuk lalu pilih Tambahkan Aturan. Grup keamanan ini akan mengizinkan akses bagi anggota dari grup keamanan lain.
4. Dari Jenis, pilih Aturan TCP Kustom.
 - a. Untuk Rentang Port, tentukan port yang Anda gunakan saat membuat klaster.

Port default untuk cluster MemoryDB adalah. 6379
 - b. Pada kotak Sumber, masukkan ID dari grup keamanan. Dari daftar, pilih grup keamanan yang akan Anda gunakan untuk instans Amazon EC2 Anda.
5. Pilih Simpan jika selesai.

Setelah Anda mengaktifkan akses, Anda sekarang siap untuk terhubung ke cluster, seperti yang dibahas di bagian berikutnya.

Untuk informasi tentang mengakses klaster MemoryDB Anda dari VPC Amazon yang berbeda, AWS Wilayah yang berbeda, atau bahkan jaringan perusahaan Anda, lihat berikut ini:

- [Pola Akses untuk Mengakses Cluster MemoryDB di Amazon VPC](#)
- [Mengakses sumber daya MemoryDB dari luar AWS](#)

Langkah 3: Connect ke cluster

Sebelum melanjutkan, selesaikan [Langkah 2: Otorisasi akses ke cluster](#).

Bagian ini mengasumsikan bahwa Anda telah membuat instans Amazon EC2 dan dapat terhubung ke instans tersebut. Untuk petunjuk cara melakukannya, lihat [Panduan Memulai Amazon EC2](#).

Instans Amazon EC2 dapat terhubung ke klaster hanya jika Anda telah mengotorisasi untuk melakukannya.

Temukan titik akhir klaster Anda

Ketika klaster dalam status tersedia dan Anda telah memberikan otorisasi akses ke klaster, Anda dapat masuk ke instans Amazon EC2 dan terhubung ke klaster tersebut. Untuk melakukan itu, Anda perlu menentukan titik akhir terlebih dahulu.

Untuk mengeksplorasi lebih lanjut cara menemukan titik akhir Anda, lihat yang berikut ini:

- [Menemukan Endpoint untuk Cluster MemoryDB \(AWS Management Console\)](#)
- [Menemukan Titik Akhir untuk Cluster MemoryDB \(CLI\)AWS](#)
- [Menemukan Endpoint untuk Cluster MemoryDB \(MemoryDB API\)](#)

Connect ke cluster MemoryDB (Linux)

Sekarang Anda memiliki titik akhir yang Anda butuhkan, Anda dapat masuk ke instans EC2 dan terhubung ke cluster. Dalam contoh berikut, Anda menggunakan utilitas cli untuk terhubung ke cluster menggunakan Ubuntu 22. Versi terbaru cli juga mendukung SSL/TLS untuk menghubungkan kluster yang diaktifkan enkripsi/otentikasi.

Menghubungkan ke node MemoryDB menggunakan redis-cli

Untuk mengakses data dari node MemoryDB, Anda menggunakan klien yang bekerja dengan Secure Socket Layer (SSL). Anda juga dapat menggunakan redis-cli dengan TLS/SSL di Amazon Linux dan Amazon Linux 2.

Untuk menggunakan redis-cli untuk terhubung ke cluster MemoryDB di Amazon Linux 2 atau Amazon Linux

1. Unduh dan kompilasi utilitas redis-cli. Utilitas ini termasuk dalam distribusi perangkat lunak Redis OSS.
2. Pada prompt perintah instans EC2 Anda, ketikkan perintah yang sesuai untuk versi Linux yang Anda gunakan.

Amazon Linux 2023

Jika menggunakan Amazon Linux 2023, masukkan ini:

```
sudo yum install redis6 -y
```

Kemudian ketik perintah berikut, ganti titik akhir cluster dan port Anda dengan apa yang ditampilkan dalam contoh ini.

```
redis-cli -h Primary or Configuration Endpoint --tls -p 6379
```

Untuk informasi selengkapnya tentang cara mencari titik akhir, lihat [Menemukan Titik Akhir Simpul Anda](#).

Amazon Linux 2

Jika menggunakan Amazon Linux 2, masukkan ini:

```
sudo yum -y install openssl-devel gcc
wget http://download.redis.io/redis-stable.tar.gz
tar xvzf redis-stable.tar.gz
cd redis-stable
make distclean
make redis-cli BUILD_TLS=yes
sudo install -m 755 src/redis-cli /usr/local/bin/
```

Amazon Linux

Jika menggunakan Amazon Linux, masukkan ini:

```
sudo yum install gcc jemalloc-devel openssl-devel tcl tcl-devel clang wget
wget http://download.redis.io/redis-stable.tar.gz
```

```
tar xvzf redis-stable.tar.gz
cd redis-stable
make redis-cli CC=clang BUILD_TLS=yes
sudo install -m 755 src/redis-cli /usr/local/bin/
```

Di Amazon Linux, Anda mungkin perlu menjalankan langkah tambahan berikut:

```
sudo yum install clang
CC=clang make
sudo make install
```

3. Setelah Anda mengunduh dan menginstal utilitas redis-cli, Anda disarankan untuk menjalankan perintah opsional. `make -test`
4. Untuk terhubung ke cluster dengan enkripsi dan otentikasi diaktifkan, masukkan perintah ini:

```
redis-cli -h Primary or Configuration Endpoint --tls -a 'your-password' -p 6379
```

Note

Jika Anda menginstal redis6 di Amazon Linux 2023, Anda sekarang dapat menggunakan `redis6-cli` perintah alih-alih: `redis-cli`

```
redis6-cli -h Primary or Configuration Endpoint --tls -p 6379
```

Langkah 4: Menghapus cluster

Selama kluster dalam status tersedia, Anda akan dikenai biaya, terlepas dari apakah Anda secara aktif menggunakannya atau tidak. Untuk menghentikan biaya, hapus kluster tersebut.

Warning

Saat Anda menghapus cluster MemoryDB, snapshot manual Anda dipertahankan. Anda juga dapat membuat snapshot akhir sebelum kluster dihapus. Snapshot otomatis tidak dipertahankan. Untuk informasi selengkapnya, lihat [Melakukan snapshot dan pemulihan](#).

Menggunakan AWS Management Console

Prosedur berikut menghapus satu klaster dari deployment Anda. Untuk menghapus beberapa klaster, ulangi prosedur untuk setiap klaster yang ingin dihapus. Anda tidak perlu menunggu satu klaster selesai dihapus sebelum memulai prosedur untuk menghapus klaster lain.

Untuk menghapus klaster

1. [Masuk ke AWS Management Console dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/.](https://console.aws.amazon.com/memorydb/)
2. Untuk memilih cluster yang akan dihapus, pilih tombol radio di sebelah nama cluster dari daftar cluster. Dalam kasus ini, nama klaster yang Anda buat di [Langkah 1: Buat cluster](#).
3. Untuk Tindakan, pilih Hapus.
4. Pertama pilih apakah akan membuat snapshot cluster sebelum menghapusnya dan kemudian masukkan delete di kotak konfirmasi dan Hapus untuk menghapus cluster, atau pilih Batal untuk menjaga cluster.

Jika Anda memilih Hapus, status klaster berubah menjadi menghapus.

Segera setelah klaster Anda tidak lagi tercantum dalam daftar klaster, Anda berhenti dikenai biaya untuk klaster tersebut.

Menggunakan AWS CLI

Kode berikut menghapus klaster `my-cluster`. Dalam kasus ini, ganti `my-cluster` dengan nama klaster yang Anda buat di [Langkah 1: Buat cluster](#).

```
aws memorydb delete-cluster --cluster-name my-cluster
```

Operasi `delete-cluster` CLI hanya menghapus satu cluster. Untuk menghapus beberapa cluster, panggil `delete-cluster` setiap cluster yang ingin Anda hapus. Anda tidak perlu menunggu satu cluster selesai menghapus sebelum menghapus yang lain.

Untuk Linux, macOS, atau Unix:

```
aws memorydb delete-cluster \  
  --cluster-name my-cluster \  
  --region us-east-1
```

Untuk Windows:

```
aws memorydb delete-cluster ^  
  --cluster-name my-cluster ^  
  --region us-east-1
```

Untuk informasi selengkapnya, lihat [delete-cluster](#).

Menggunakan MemoryDB API

Kode berikut menghapus klaster `my-cluster`. Dalam kasus ini, ganti `my-cluster` dengan nama klaster yang Anda buat di [Langkah 1: Buat cluster](#).

```
https://memory-db.us-east-1.amazonaws.com/  
?Action>DeleteCluster  
&ClusterName=my-cluster  
&Region=us-east-1  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20210802T220302Z  
&X-Amz-Algorithm=Amazon4-HMAC-SHA256  
&X-Amz-Date=20210802T220302Z  
&X-Amz-SignedHeaders=Host  
&X-Amz-Expires=20210802T220302Z  
&X-Amz-Credential=<credential>  
&X-Amz-Signature=<signature>
```

Operasi `DeleteCluster` API hanya menghapus satu cluster. Untuk menghapus beberapa cluster, panggil `DeleteCluster` setiap cluster yang ingin Anda hapus. Anda tidak perlu menunggu satu cluster selesai menghapus sebelum menghapus yang lain.

Untuk informasi lebih lanjut, lihat [DeleteCluster](#).

Apa yang saya lakukan selanjutnya?

Sekarang setelah Anda mencoba latihan Memulai, Anda dapat menjelajahi bagian berikut untuk mempelajari lebih lanjut tentang MemoryDB dan alat yang tersedia:

- [Memulai dengan AWS](#)
- [Alat untuk Amazon Web Services](#)

- [AWS Antarmuka Baris Perintah](#)
- [Referensi API MemoryDB.](#)

Mengelola simpul

Node adalah blok bangunan terkecil dari penyebaran MemoryDB. Sebuah simpul milik pecahan yang termasuk dalam cluster. Setiap node menjalankan versi mesin yang dipilih saat cluster dibuat atau terakhir dimodifikasi. Setiap simpul memiliki nama dan port Layanan Nama Domain (DNS) sendiri. Beberapa jenis node MemoryDB didukung, masing-masing dengan berbagai jumlah memori terkait dan daya komputasi.

Topik

- [Node dan pecahan MemoryDB](#)
- [Jenis simpul yang didukung](#)
- [Node cadangan MemoryDB](#)
- [Mengganti simpul](#)

Beberapa operasi penting yang berkaitan dengan simpul adalah sebagai berikut:

- [Menambahkan/Menghapus node dari cluster](#)
- [Penskalaan](#)
- [Menemukan titik akhir koneksi](#)

Node dan pecahan MemoryDB

Shard adalah susunan hierarkis node, masing-masing dibungkus dalam sebuah cluster. Serpihan mendukung replikasi. Di dalam serpihan, satu simpul berfungsi sebagai simpul primer baca/tulis. Semua simpul lain dalam serpihan berfungsi sebagai replika hanya-baca dari simpul primer. MemoryDB mendukung beberapa pecahan dalam sebuah cluster. Dukungan ini memungkinkan partisi data Anda dalam cluster MemoryDB.

MemoryDB mendukung replikasi melalui pecahan. Operasi API [DescribeClusters](#) mencantumkan pecahan dengan node anggota, nama node, titik akhir dan juga informasi lainnya.

Setelah cluster MemoryDB dibuat, itu dapat diubah (diskalakan masuk atau keluar). Untuk informasi lain, lihat [Penskalaan](#) dan [Mengganti simpul](#).

Ketika Anda membuat klaster baru, Anda dapat melakukan seeding klaster tersebut dengan data dari klaster lama sehingga klaster baru tidak dimulai dari kosong. Melakukan hal ini dapat membantu

jika Anda perlu mengubah jenis node, versi mesin, atau bermigrasi dari Amazon ElastiCache (Redis OSS). Untuk informasi selengkapnya, lihat [Membuat snapshot manual](#) dan [Memulihkan dari snapshot](#).

Jenis simpul yang didukung

MemoryDB mendukung jenis node berikut.

Memori yang dioptimalkan

Jenis instans	Bandwidth acuan (Gbps)	Bandwidth lonjakan (Gbps)	Peningkatan I/O Multiplexing (Redis OSS 7.0.4+)	Versi mesin minimum
db.r7g.large	0,937	12,5	Tidak	6.2
db.r7g.xlarge	1.876	12,5	Tidak	6.2
db.r7g.2xlarge	3,75	15	Ya	6.2
db.r7g.4xlarge	7,5	15	Ya	6.2
db.r7g.8xlarge	15	N/A	Ya	6.2
db.r7g.12xlarge	22.5	N/A	Ya	6.2
db.r7g.16xlarge	30	N/A	Ya	6.2
db.r6g.large	0,75	10.0	Tidak	6.2
db.r6g.xlarge	1,25	10.0	Tidak	6.2
db.r6g.2xlarge	2.5	10.0	Ya	6.2
db.r6g.4xlarge	5.0	10.0	Ya	6.2
db.r6g.8xlarge	12	N/A	Ya	6.2
db.r6g.12xlarge	20	N/A	Ya	6.2
db.r6g.16xlarge	25	N/A	Ya	6.2

Memori yang dioptimalkan dengan tingkatan data

Jenis instans	Bandwidth acuan (Gbps)	Bandwidth lonjakan (Gbps)	Peningkatan I/O Multiplexing (Redis OSS 7.0.4+)	Versi mesin minimum
db.r6gd.xlarge	1,25	10	Tidak	6.2
db.r6gd.2xlarge	2.5	10	Tidak	6.2
db.r6gd.4xlarge	5.0	10	Tidak	6.2
db.r6gd.8xlarge	12	N/A	Tidak	6.2

Node tujuan umum

Jenis instans	Bandwidth acuan (Gbps)	Bandwidth lonjakan (Gbps)	Peningkatan I/O Multiplexing (Redis OSS 7.0.4+)	Versi mesin minimum
db.t4g.small	0,18	5.0	Tidak	6.2
db.t4g.medium	0,256	5.0	Tidak	6.2

Untuk ketersediaan AWS Wilayah, lihat Harga [MemoryDB](#)

Semua jenis node dibuat dalam virtual private cloud (VPC).

Node cadangan MemoryDB

Node cadangan memberi Anda diskon yang signifikan dibandingkan dengan harga node sesuai permintaan. Node yang dicadangkan bukanlah node fisik, melainkan diskon penagihan yang diterapkan pada penggunaan node sesuai permintaan di akun Anda. Diskon untuk node yang dipesan terkait dengan tipe node dan AWS Wilayah.

Proses umum untuk bekerja dengan node yang dicadangkan adalah sebagai berikut:

- Tinjau informasi tentang penawaran node cadangan yang tersedia
- Beli penawaran node cadangan menggunakan AWS Management Console, AWS Command Line Interface atau SDK
- Tinjau informasi tentang node cadangan Anda yang ada

Topik

- [Ikhtisar node yang dicadangkan](#)

Ikhtisar node yang dicadangkan

Ketika Anda membeli node cadangan MemoryDB, Anda membeli komitmen untuk mendapatkan tarif diskon, pada jenis node tertentu, selama durasi node yang dicadangkan. Untuk menggunakan node cadangan MemoryDB, Anda membuat node baru seperti yang Anda lakukan untuk node sesuai permintaan. Node baru yang Anda buat harus sesuai dengan spesifikasi node yang dicadangkan. Jika spesifikasi node baru cocok dengan node cadangan yang ada untuk akun Anda, Anda akan ditagih dengan tarif diskon yang ditawarkan untuk node yang dicadangkan. Jika tidak, node ditagih pada tingkat permintaan. Anda dapat menggunakan API AWS Management Console, AWS CLI, atau MemoryDB untuk membuat daftar dan membeli penawaran node cadangan yang tersedia.

MemoryDB menawarkan node cadangan untuk memori yang dioptimalkan R7g, R6g, dan R6gd (dengan data tiering) node. Untuk informasi harga, lihat Harga [MemoryDB](#).

Jenis penawaran

Node cadangan tersedia dalam tiga varietas — No Upfront, Partial Upfront, dan All Upfront — yang memungkinkan Anda mengoptimalkan biaya MemoryDB berdasarkan penggunaan yang Anda harapkan.

No Upfront — Opsi ini menyediakan akses ke node yang dipesan tanpa memerlukan pembayaran di muka. Node cadangan No Upfront Anda menagih tarif per jam diskon untuk setiap jam dalam jangka waktu tersebut, terlepas dari penggunaannya, dan tidak diperlukan pembayaran di muka.

Partial Upfront — Opsi ini membutuhkan bagian dari node yang dicadangkan untuk dibayar di muka. Sisa jam dalam jangka waktu pemesanan akan ditagih dengan tarif per jam yang didiskon, terlepas dari penggunaannya.

Semua di muka — Pembayaran penuh dilakukan pada awal jangka waktu, tanpa biaya lain yang dikeluarkan untuk sisa jangka waktu terlepas dari jumlah jam yang digunakan.

Ketiga jenis penawaran tersedia dalam jangka waktu satu tahun dan tiga tahun.

Ukuran node cadangan fleksibel

Ketika Anda membeli node reserved, satu hal yang Anda tentukan adalah tipe node, misalnya db.r6g.xlarge. Untuk informasi selengkapnya, tentang jenis node, lihat Harga [MemoryDB](#).

Jika Anda memiliki node, dan Anda perlu menskalakannya ke kapasitas yang lebih besar, node cadangan Anda secara otomatis diterapkan ke node skala Anda. Artinya, node cadangan Anda secara otomatis diterapkan untuk penggunaan ukuran apa pun dalam keluarga simpul yang sama. Node cadangan fleksibel ukuran tersedia untuk node dengan Wilayah yang sama. AWS Node cadangan fleksibel ukuran hanya dapat menskalakan dalam keluarga simpulnya. Misalnya, node cadangan untuk db.r6g.xlarge dapat diterapkan ke db.r6g.2xlarge, tetapi tidak ke db.r6gd.large, karena db.r6g dan db.r6gd adalah keluarga simpul yang berbeda.

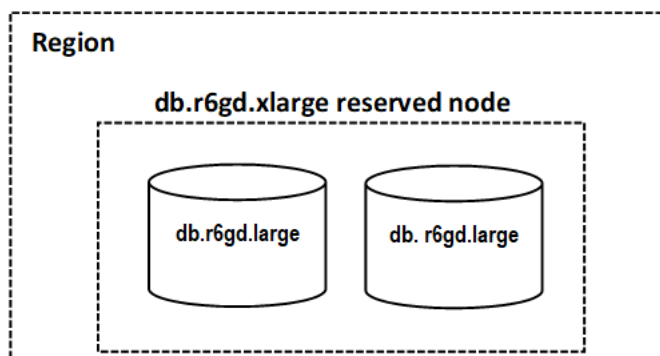
Fleksibilitas ukuran berarti Anda dapat bergerak bebas di antara konfigurasi dalam keluarga simpul yang sama. Misalnya, Anda dapat berpindah dari node cadangan r6g.xlarge (8 unit dinormalisasi) ke dua r6g.large node cadangan (8 unit dinormalisasi) ($2 \times 4 = 8$ unit dinormalisasi) di Wilayah yang sama tanpa biaya tambahan. AWS

Anda dapat membandingkan penggunaan untuk ukuran node cadangan yang berbeda dengan menggunakan unit yang dinormalisasi. Misalnya, satu unit penggunaan pada dua node db.r6g.4xlarge setara dengan 16 unit penggunaan yang dinormalisasi pada satu db.r6g.large. Tabel berikut menunjukkan jumlah unit dinormalisasi untuk setiap ukuran node:

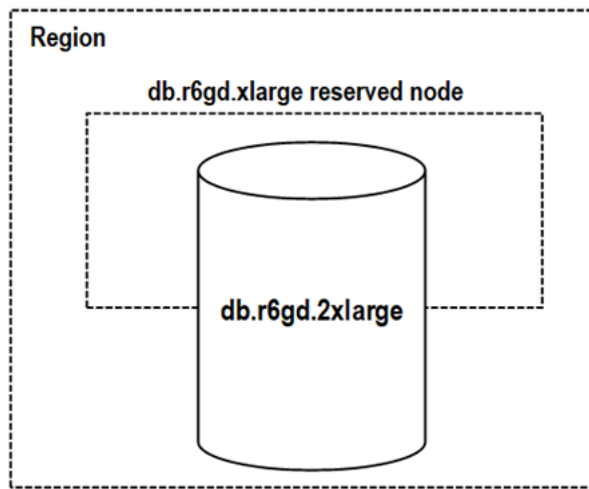
Ukuran simpul	Unit yang dinormalisasi
small	1

Ukuran simpul	Unit yang dinormalisasi
medium	2
large	4
xlarge	8
2xlarge	16
4xlarge	32
6xlarge	48
8xlarge	64
10xlarge	80
12xlarge	96
16xlarge	128

Misalnya, Anda membeli node cadangan `db.r6gd.xlarge`, dan Anda memiliki dua node cadangan `db.r6gd.large` yang sedang berjalan di akun Anda di Wilayah yang sama. AWS Dalam hal ini, manfaat penagihan diterapkan secara penuh ke kedua node.



Atau, jika Anda memiliki satu instans `db.r6gd.2xlarge` yang berjalan di akun Anda di AWS Wilayah yang sama, manfaat penagihan diterapkan ke 50 persen dari penggunaan node yang dicadangkan.



Menghapus node yang dicadangkan

Persyaratan untuk node cadangan melibatkan komitmen satu tahun atau tiga tahun. Anda tidak dapat membatalkan node yang dicadangkan. Namun, Anda dapat menghapus node yang dicakup oleh discount node reserved. Proses untuk menghapus node yang dicakup oleh discount node reserved sama dengan node lainnya.

Jika Anda menghapus node yang dicakup oleh discount node reserved, Anda dapat meluncurkan node lain dengan spesifikasi yang kompatibel. Dalam hal ini, Anda tetap mendapatkan tarif diskon selama jangka waktu pemesanan (satu atau tiga tahun).

Bekerja dengan node yang dicadangkan

Anda dapat menggunakan API AWS Management Console, the AWS Command Line Interface, dan MemoryDB untuk bekerja dengan node yang dicadangkan.

Konsol

Untuk mendapatkan harga dan informasi tentang penawaran node cadangan yang tersedia

1. [Masuk ke AWS Management Console dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/.](https://console.aws.amazon.com/memorydb/)
2. Di panel navigasi, pilih Node cadangan.
3. Pilih Beli node yang dicadangkan.
4. Untuk tipe Node, pilih jenis node yang ingin Anda gunakan.
5. Untuk Kuantitas, pilih jumlah node yang ingin Anda gunakan.

6. Untuk Term, pilih lamanya waktu yang Anda inginkan node database dicadangkan.
7. Untuk Jenis penawaran, pilih jenis penawaran.

Setelah Anda membuat pilihan ini, Anda dapat melihat informasi harga di bawah Ringkasan reservasi.

 Important

Pilih Batal untuk menghindari pembelian node yang dipesan ini dan menimbulkan biaya apa pun.

Setelah Anda memiliki informasi tentang penawaran node cadangan yang tersedia, Anda dapat menggunakan informasi tersebut untuk membeli penawaran seperti yang ditunjukkan dalam prosedur berikut:

Untuk membeli node yang dicadangkan

1. [Masuk ke AWS Management Console dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/.](https://console.aws.amazon.com/memorydb/)
2. Di panel navigasi, pilih Node cadangan.
3. Pilih Beli node yang dicadangkan.
4. Untuk tipe Node, pilih jenis node yang ingin Anda gunakan.
5. Untuk Kuantitas, pilih jumlah node yang ingin Anda gunakan.
6. Untuk Term, pilih lamanya waktu yang Anda inginkan node database dicadangkan.
7. Untuk Jenis penawaran, pilih jenis penawaran.
8. (Opsional) Anda dapat menetapkan pengenalan Anda sendiri ke node cadangan yang Anda beli untuk membantu Anda melacaknya. Untuk ID Reservasi, ketik pengenalan untuk node reservasi Anda.

Setelah Anda membuat pilihan ini, Anda dapat melihat informasi harga di bawah Ringkasan reservasi.

9. Pilih Beli node yang dicadangkan.
10. Node cadangan Anda dibeli, lalu ditampilkan dalam daftar Node Cadangan.

Untuk mendapatkan informasi tentang node yang dicadangkan untuk AWS akun Anda

1. [Masuk ke AWS Management Console dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/.](https://console.aws.amazon.com/memorydb/)
2. Di panel navigasi, pilih Node cadangan.
3. Node yang dicadangkan untuk akun Anda muncul. Untuk melihat informasi rinci tentang node reserved tertentu, pilih node tersebut dalam daftar. Anda kemudian dapat melihat informasi rinci tentang node itu secara detail.

AWS Command Line Interface

`describe-reserved-nodes-offerings` Contoh berikut mengembalikan rincian penawaran reserved-node.

```
aws memorydb describe-reserved-nodes-offerings
```

Ini menghasilkan output yang mirip dengan yang berikut:

```
{
  "ReservedNodesOfferings": [
    {
      "ReservedNodesOfferingId": "0193cc9d-7037-4d49-b332-xxxxxxxxxxxx",
      "NodeType": "db.xxx.large",
      "Duration": 94608000,
      "FixedPrice": $xxx.xx,
      "OfferingType": "Partial Upfront",
      "RecurringCharges": [
        {
          "RecurringChargeAmount": $xx.xx,
          "RecurringChargeFrequency": "Hourly"
        }
      ]
    }
  ]
}
```

Anda juga dapat meneruskan parameter berikut untuk membatasi ruang lingkup dari apa yang dikembalikan:

- `--reserved-nodes-offering-id` – ID penawaran yang ingin Anda beli.
- `--node-type`— Nilai filter tipe node. Gunakan parameter ini untuk menampilkan hanya reservasi yang cocok dengan tipe node yang ditentukan.
- `--duration`— Nilai filter durasi, ditentukan dalam tahun atau detik. Gunakan parameter ini untuk hanya menampilkan reservasi selama durasi ini.
- `--offering-type`— Gunakan parameter ini untuk hanya menampilkan penawaran yang tersedia yang cocok dengan jenis penawaran yang ditentukan.

Setelah Anda memiliki informasi tentang penawaran node cadangan yang tersedia, Anda dapat menggunakan informasi tersebut untuk membeli penawaran.

`purchase-reserved-nodes-offering` Contoh berikut membeli node cadangan baru

Untuk Linux, macOS, atau Unix:

```
aws memorydb purchase-reserved-nodes-offering \
  --reserved-nodes-offering-id 0193cc9d-7037-4d49-b332-d5e984f1d8ca \
  --reservation-id reservation \
  --node-count 2
```

Untuk Windows:

```
aws memorydb purchase-reserved-nodes-offering ^
  --reserved-nodes-offering-id 0193cc9d-7037-4d49-b332-d5e984f1d8ca ^
  --reservation-id MyReservation
```

- `--reserved-nodes-offering-id` mewakili nama node cadangan yang menawarkan untuk membeli.
- `--reservation-id` adalah pengenal yang ditentukan pelanggan untuk melacak reservasi ini.

Note

ID Reservasi adalah pengenal unik yang ditentukan pelanggan untuk melacak reservasi ini. Jika parameter ini tidak ditentukan, MemoryDB secara otomatis menghasilkan pengenal untuk reservasi.

- `--node-count` adalah jumlah node yang akan dicadangkan. Defaultnya ke 1.

Ini menghasilkan output yang mirip dengan yang berikut:

```
{
  "ReservedNode": {
    "ReservationId": "reservation",
    "ReservedNodesOfferingId": "0193cc9d-7037-4d49-b332-xxxxxxxxxxxx",
    "NodeType": "db.xxx.large",
    "StartTime": 1671173133.982,
    "Duration": 94608000,
    "FixedPrice": $xxx.xx,
    "NodeCount": 2,
    "OfferingType": "Partial Upfront",
    "State": "payment-pending",
    "RecurringCharges": [
      {
        "RecurringChargeAmount": $xx.xx,
        "RecurringChargeFrequency": "Hourly"
      }
    ],
    "ARN": "arn:aws:memorydb:us-east-1:xxxxxxx:reservednode/reservation"
  }
}
```

Setelah Anda membeli node yang dicadangkan, Anda bisa mendapatkan informasi tentang node cadangan Anda.

`describe-reserved-nodes` Contoh berikut mengembalikan informasi tentang node yang dicadangkan untuk akun ini.

```
aws memorydb describe-reserved-nodes
```

Ini menghasilkan output yang mirip dengan yang berikut:

```
{
  "ReservedNodes": [
    {
      "ReservationId": "ri-2022-12-16-00-28-40-600",
      "ReservedNodesOfferingId": "0193cc9d-7037-4d49-b332-xxxxxxxxxxxx",
      "NodeType": "db.xxx.large",
      "StartTime": 1671150737.969,
```

```

    "Duration": 94608000,
    "FixedPrice": $xxx.xx,
    "NodeCount": 1,
    "OfferingType": "Partial Upfront",
    "State": "active",
    "RecurringCharges": [
      {
        "RecurringChargeAmount": $xx.xx,
        "RecurringChargeFrequency": "Hourly"
      }
    ],
    "ARN": "arn:aws:memorydb:us-east-1:xxxxxxx:reservednode/ri-2022-12-16-00-28-40-600"
  }
]
}

```

Anda juga dapat meneruskan parameter berikut untuk membatasi ruang lingkup dari apa yang dikembalikan:

- `--reservation-id`— Anda dapat menetapkan pengenal Anda sendiri ke node cadangan yang Anda beli untuk membantu melacaknya.
- `--reserved-nodes-offering-id`— Nilai filter pengenal penawaran. Gunakan parameter ini untuk menampilkan hanya reservasi yang dibeli yang cocok dengan pengenal penawaran yang ditentukan.
- `--node-type`— Nilai filter tipe node. Gunakan parameter ini untuk menampilkan hanya reservasi yang cocok dengan tipe node yang ditentukan.
- `--duration`— Nilai filter durasi, ditentukan dalam tahun atau detik. Gunakan parameter ini untuk hanya menampilkan reservasi selama durasi ini.
- `--offering-type`— Gunakan parameter ini untuk hanya menampilkan penawaran yang tersedia yang cocok dengan jenis penawaran yang ditentukan.

MemoryDB API

Contoh berikut menunjukkan cara menggunakan [MemoryDB Query API](#) untuk node yang dicadangkan:

DescribeReservedNodesOfferings

Mengembalikan rincian penawaran reserved-node.

```
https://memorydb.us-west-2.amazonaws.com/  
  ?Action=DescribeReservedNodesOfferings  
  &ReservedNodesOfferingId=649fd0c8-xxxx-xxxx-xxxx-06xxxx75e95f  
&"Duration": 94608000,  
  &NodeType="db.r6g.large"  
  &OfferingType="Partial Upfront"  
  &Version=2021-01-01  
  &SignatureVersion=4  
  &SignatureMethod=HmacSHA256  
  &Timestamp=20141201T220302Z  
  &X-Amz-Algorithm  
  &X-Amz-SignedHeaders=Host  
  &X-Amz-Expires=20141201T220302Z  
  &X-Amz-Credential=<credential>  
  &X-Amz-Signature=<signature>
```

Parameter berikut membatasi ruang lingkup dari apa yang dikembalikan:

- `ReservedNodesOfferingId` mewakili nama node cadangan yang menawarkan untuk membeli.
- `Duration`— Nilai filter durasi, ditentukan dalam tahun atau detik. Gunakan parameter ini untuk hanya menampilkan reservasi selama durasi ini.
- `NodeType`— Nilai filter tipe node. Gunakan parameter ini untuk menampilkan hanya penawaran yang cocok dengan tipe node yang ditentukan.
- `OfferingType`— Gunakan parameter ini untuk hanya menampilkan penawaran yang tersedia yang cocok dengan jenis penawaran yang ditentukan.

Setelah Anda memiliki informasi tentang penawaran node cadangan yang tersedia, Anda dapat menggunakan informasi tersebut untuk membeli penawaran.

PurchaseReservedNodesOffering

Memungkinkan Anda membeli penawaran node yang dicadangkan.

```
https://memorydb.us-west-2.amazonaws.com/  
  ?Action=PurchaseReservedCacheNodesOffering  
  &ReservedNodesOfferingId=649fd0c8-xxxx-xxxx-xxxx-06xxxx75e95f  
  &ReservationID=myreservationID  
  &NodeCount=1  
  &Version=2021-01-01
```

```

&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20141201T220302Z
&X-Amz-Algorithm
&X-Amz-SignedHeaders=Host
&X-Amz-Expires=20141201T220302Z
&X-Amz-Credential=<credential>
&X-Amz-Signature=<signature>

```

- `ReservedNodesOfferingId` mewakili nama node cadangan yang menawarkan untuk membeli.
- `ReservationID` adalah pengenal yang ditentukan pelanggan untuk melacak reservasi ini.

Note

ID Reservasi adalah pengenal unik yang ditentukan pelanggan untuk melacak reservasi ini. Jika parameter ini tidak ditentukan, MemoryDB secara otomatis menghasilkan pengenal untuk reservasi.

- `NodeCount` adalah jumlah node yang akan dicadangkan. Defaultnya ke 1.

Setelah Anda membeli node yang dicadangkan, Anda bisa mendapatkan informasi tentang node cadangan Anda.

DescribeReservedNodes

Mengembalikan informasi tentang node yang dicadangkan untuk akun ini.

```

https://memorydb.us-west-2.amazonaws.com/
?Action=DescribeReservedNodes
&ReservedNodesOfferingId=649fd0c8-xxxx-xxxx-xxxx-06xxxx75e95f
&ReservationID=myreservationID
&NodeType="db.r6g.large"
&Duration=94608000
&OfferingType="Partial Upfront"
&Version=2021-01-01
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20141201T220302Z
&X-Amz-Algorithm
&X-Amz-SignedHeaders=Host
&X-Amz-Expires=20141201T220302Z

```

```
&X-Amz-Credential=<credential>
&X-Amz-Signature=<signature>
```

Parameter berikut membatasi ruang lingkup dari apa yang dikembalikan:

- **ReservedNodesOfferingId** mewakili nama node reserved.
- **ReservationID**— Anda dapat menetapkan pengenal Anda sendiri ke node cadangan yang Anda beli untuk membantu melacaknya.
- **NodeType**— Nilai filter tipe node. Gunakan parameter ini untuk menampilkan hanya reservasi yang cocok dengan tipe node yang ditentukan.
- **Duration**— Nilai filter durasi, ditentukan dalam tahun atau detik. Gunakan parameter ini untuk hanya menampilkan reservasi selama durasi ini.
- **OfferingType**— Gunakan parameter ini untuk hanya menampilkan penawaran yang tersedia yang cocok dengan jenis penawaran yang ditentukan.

Melihat tagihan untuk node yang Anda pesan

Anda dapat melihat penagihan untuk node yang dipesan di Dasbor Penagihan di AWS Management Console

Untuk melihat penagihan node yang dipesan

1. [Masuk ke AWS Management Console dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/.](https://console.aws.amazon.com/memorydb/)
2. Dari tombol Cari di bagian atas konsol, pilih Penagihan.
3. Pilih Tagihan dari sisi kiri dasbor.
4. Di bawah Biaya AWS Layanan, perluas MemoryDB.
5. Perluas AWS Wilayah tempat node cadangan Anda berada, misalnya US East (Virginia N.).

Node cadangan Anda dan biaya per jamnya untuk bulan berjalan ditampilkan di bawah Instans Cadangan Amazon MemoryDB CreateCluster .

Amazon MemoryDB CreateCluster Reserved Instances		
AmazonMemoryDB, db.r6g.large reserved instance applied	81.000 Hrs	
AmazonMemoryDB, db.r6g.4xlarge reserved instance applied	324.000 Hrs	
AmazonMemoryDB, db.r6g.4xlarge reserved instance applied	162.000 Hrs	
USD hourly fee per AmazonMemoryDB, db.r6g.large instance	1,488.000 Hrs	
USD hourly fee per AmazonMemoryDB, db.r6gd.2xlarge instance	744.000 Hrs	
USD hourly fee per AmazonMemoryDB, db.r6g.4xlarge instance	744.000 Hrs	
USD hourly fee per AmazonMemoryDB, db.r6gd.xlarge instance	744.000 Hrs	
USD hourly fee per AmazonMemoryDB, db.r6gd.4xlarge instance	2,976.000 Hrs	

Mengganti simpul

MemoryDB sering meningkatkan armadanya dengan tambalan dan peningkatan, biasanya mulus. Namun, dari waktu ke waktu kami perlu meluncurkan kembali node MemoryDB Anda untuk menerapkan pembaruan OS wajib ke host yang mendasarinya. Penggantian ini diperlukan untuk menerapkan peningkatan yang memperkuat keamanan, keandalan, dan performa operasional.

Anda memiliki opsi untuk mengelola penggantian ini sendiri setiap saat sebelum periode penggantian simpul yang terjadwal. Ketika Anda mengelola penggantian sendiri, instans Anda menerima pembaruan OS ketika Anda meluncurkan kembali simpul tersebut dan penggantian simpul terjadwal Anda dibatalkan. Anda mungkin akan terus menerima peringatan yang menunjukkan bahwa penggantian simpul harus dilakukan. Jika Anda telah mengurangi kebutuhan pemeliharaan secara manual, Anda dapat mengabaikan peringatan ini.

Note

Node pengganti yang dihasilkan secara otomatis oleh MemoryDB mungkin memiliki alamat IP yang berbeda. Anda bertanggung jawab untuk meninjau konfigurasi aplikasi Anda untuk memastikan bahwa node Anda terkait dengan alamat IP yang sesuai.

Daftar berikut mengidentifikasi tindakan yang dapat Anda lakukan saat MemoryDB menjadwalkan salah satu node Anda untuk diganti:

Opsi penggantian node MemoryDB

- Tidak melakukan apa-apa - Jika Anda tidak melakukan apa-apa, MemoryDB menggantikan node sesuai jadwal.

Jika node adalah anggota cluster multi-AZ, MemoryDB menyediakan ketersediaan yang lebih baik selama penambalan, pembaruan, dan penggantian node terkait pemeliharaan lainnya.

Penggantian selesai saat cluster melayani permintaan tulis masuk.

- Ubah jendela pemeliharaan Anda - Untuk acara pemeliharaan terjadwal, Anda menerima email atau acara pemberitahuan dari MemoryDB. Dalam hal ini, jika Anda mengubah periode pemeliharaan Anda sebelum waktu penggantian terjadwal, simpul Anda sekarang akan diganti pada waktu yang baru. Untuk informasi selengkapnya, lihat [Memodifikasi cluster MemoryDB](#).

Note

Kemampuan untuk mengubah jendela pengganti Anda dengan memindahkan jendela pemeliharaan Anda hanya tersedia ketika pemberitahuan MemoryDB menyertakan jendela pemeliharaan. Jika notifikasi tersebut tidak menyertakan periode pemeliharaan, Anda tidak dapat mengubah periode pengganti.

Misalnya, katakanlah pemeliharaan dilakukan pada Kamis, 9 November, pukul 15.00 dan periode pemeliharaan berikutnya adalah Jumat, 10 November, pukul 17.00. Berikut adalah tiga skenario dengan hasilnya:

- Anda mengubah periode pemeliharaan Anda ke Jumat pukul 16.00, setelah tanggal dan waktu saat ini dan sebelum periode pemeliharaan terjadwal berikutnya. Simpul diganti pada hari Jumat, 10 November, pukul 16.00.
- Anda mengubah periode pemeliharaan Anda ke Sabtu pukul 16.00, setelah tanggal dan waktu saat ini dan sebelum periode pemeliharaan terjadwal berikutnya. Simpul diganti pada hari Sabtu, 11 November, pukul 16.00.
- Anda mengubah jendela pemeliharaan Anda menjadi Rabu pukul 16:00, lebih awal dalam seminggu dari tanggal dan waktu saat ini. Simpul akan diganti pada Rabu depan, 15 November, pukul 16.00.

Untuk petunjuk, lihat [Mengelola pemeliharaan](#).

Mengelola klaster

Sebagian besar operasi MemoryDB dilakukan di tingkat cluster. Anda dapat menyiapkan klaster dengan jumlah simpul tertentu dan grup parameter yang mengontrol properti untuk setiap simpul. Semua simpul dalam klaster dirancang agar berupa jenis simpul yang sama serta memiliki parameter dan pengaturan grup keamanan yang sama.

Setiap klaster harus memiliki pengidentifikasi klaster. Pengidentifikasi klaster adalah nama yang diberikan pelanggan untuk klaster. Identifier ini menentukan cluster tertentu ketika berinteraksi dengan MemoryDB API dan perintah. AWS CLI Pengidentifikasi klaster harus unik untuk pelanggan tersebut di suatu AWS Wilayah.

Cluster MemoryDB dirancang untuk diakses menggunakan instans Amazon EC2. Anda hanya dapat meluncurkan cluster MemoryDB Anda di cloud pribadi virtual (VPC) berdasarkan layanan Amazon VPC, tetapi Anda dapat mengaksesnya dari luar. AWS Untuk informasi selengkapnya, lihat [Mengakses sumber daya MemoryDB dari luar AWS](#).

Tingkatan data

Cluster yang menggunakan tipe node dari keluarga r6gd memiliki data berjenjang antara memori dan penyimpanan SSD lokal (solid state drive). Tiering data menyediakan opsi harga-kinerja baru untuk beban kerja Redis OSS dengan memanfaatkan solid state drive (SSD) berbiaya rendah di setiap node cluster selain menyimpan data dalam memori. Mirip dengan tipe node lainnya, data yang ditulis ke node r6gd disimpan dengan tahan lama dalam log transaksi multi-AZ. Tingkatan data sangat ideal untuk beban kerja yang mengakses hingga 20 persen dari keseluruhan set datanya secara rutin, dan untuk aplikasi yang dapat menoleransi latensi tambahan saat mengakses data di SSD.

Pada cluster dengan tiering data, MemoryDB memantau waktu akses terakhir dari setiap item yang disimpannya. Ketika memori yang tersedia (DRAM) sepenuhnya dikonsumsi, MemoryDB menggunakan algoritma yang paling tidak baru digunakan (LRU) untuk secara otomatis memindahkan item yang jarang diakses dari memori ke SSD. Ketika data pada SSD kemudian diakses, MemoryDB secara otomatis dan asinkron memindahkannya kembali ke memori sebelum memproses permintaan. Jika Anda memiliki beban kerja yang mengakses hanya subset dari datanya secara teratur, tingkatan data adalah cara optimal untuk menskalakan kapasitas Anda dengan hemat biaya.

Perhatikan bahwa saat menggunakan tingkatan data, kunci itu sendiri selalu tetap dalam memori, sedangkan LRU mengatur penempatan nilai pada memori vs disk. Secara umum, sebaiknya buat kunci Anda lebih kecil dari ukuran nilai Anda saat menggunakan tingkatan data.

Tingkatan data dirancang untuk memiliki dampak performa minimal pada beban kerja aplikasi. Misalnya, dengan asumsi nilai String 500 byte, Anda biasanya dapat mengharapkan latensi 450 mikrodetik tambahan untuk permintaan baca ke data yang disimpan di SSD dibandingkan dengan permintaan baca ke data dalam memori.

Dengan ukuran node tiering data terbesar (db.r6gd.8xlarge), Anda dapat menyimpan hingga ~500 TB dalam satu cluster 500 node (250 TB saat menggunakan 1 replika baca). Untuk tiering Data, MemoryDB menyimpan 19% memori (DRAM) per node untuk penggunaan non-data. Tiering data kompatibel dengan semua perintah Redis OSS dan struktur data yang didukung di MemoryDB. Anda tidak memerlukan perubahan sisi klien untuk menggunakan fitur ini.

Topik

- [Praktik terbaik](#)
- [Batasan](#)
- [Harga tingkatan data](#)

- [Pemantauan](#)
- [Menggunakan tingkatan data](#)
- [Memulihkan data dari snapshot ke cluster dengan tiering data diaktifkan](#)

Praktik terbaik

Kami merekomendasikan praktik terbaik berikut:

- Tingkatan data sangat ideal untuk beban kerja yang mengakses hingga 20 persen dari keseluruhan set datanya secara rutin, dan untuk aplikasi yang dapat menoleransi latensi tambahan saat mengakses data di SSD.
- Saat menggunakan kapasitas SSD yang tersedia pada simpul bertingkatan data, kami menyarankan agar ukuran nilai lebih besar dari ukuran kunci. Ukuran nilai tidak boleh lebih besar dari 128MB; jika tidak, itu tidak akan dipindahkan ke disk. Ketika item dipindahkan antara DRAM dan SSD, kunci akan selalu tetap dalam memori dan hanya nilai yang dipindahkan ke tingkat SSD.

Batasan

Tingkatan data memiliki batasan berikut:

- Jenis node yang Anda gunakan harus berasal dari keluarga r6gd, yang tersedia di wilayah berikut: us-east-2,,,us-east-1,us-west-2,us-west-1,eu-west-1,eu-west-3,eu-central-1, ap-northeast-1 ap-southeast-1ap-southeast-2, ap-south-1 dan. ca-central-1 sa-east-1
- Anda tidak dapat memulihkan snapshot cluster r6gd ke cluster lain kecuali jika ia juga menggunakan r6gd.
- Anda tidak dapat mengekspor snapshot ke Amazon S3 untuk kluster tingkat data.
- Penyimpanan forkless tidak didukung.
- Penskalaan tidak didukung dari klaster tingkatan data (misalnya, sebuah klaster yang menggunakan jenis simpul r6gd) ke klaster yang tidak menggunakan tingkatan data (misalnya, klaster yang menggunakan jenis simpul r6g).
- Tingkatan data hanya mendukung kebijakan maxmemory volatile-lru, allkeys-lru, dan noeviction.
- Item yang lebih besar dari 128 MiB tidak dipindahkan ke SSD.

Harga tingkatan data

Node R6gd memiliki kapasitas total 5x lebih besar (memori+SSD) dan dapat membantu Anda mencapai penghematan biaya penyimpanan lebih dari 60 persen saat berjalan pada pemanfaatan maksimum dibandingkan dengan node R6g (hanya memori). Untuk informasi lebih lanjut, lihat harga [MemoryDB](#).

Pemantauan

MemoryDB menawarkan metrik yang dirancang khusus untuk memantau cluster kinerja yang menggunakan tingkatan data. Untuk memantau rasio item dalam DRAM dibandingkan dengan SSD, Anda dapat menggunakan CurrItems metrik di.. [Metrik untuk MemoryDB](#) Anda dapat menghitung persentase sebagai: $(\text{CurrItems with Dimension: Tier} = \text{Memory} * 100) / (\text{CurrItems with no dimension filter})$. Ketika persentase item dalam memori berkurang di bawah 5 persen, kami sarankan Anda mempertimbangkannya [Penskalaan cluster MemoryDB](#).

Untuk informasi selengkapnya, lihat Metrik untuk klaster MemoryDB yang menggunakan tingkat data di. [Metrik untuk MemoryDB](#)

Menggunakan tingkatan data

Menggunakan data tiering menggunakan AWS Management Console

Saat membuat cluster, Anda menggunakan tiering data dengan memilih tipe node dari keluarga r6gd, seperti db.r6gd.xlarge. Memilih jenis simpul tersebut secara otomatis mengaktifkan tingkatan data.

Untuk informasi tentang cara membuat klaster, lihat [Langkah 1: Buat cluster](#).

Mengaktifkan tiering data menggunakan AWS CLI

Saat membuat cluster menggunakan AWS CLI, Anda menggunakan tiering data dengan memilih tipe node dari keluarga r6gd, seperti db.r6gd.xlarge dan mengatur parameter. `--data-tiering`

Anda tidak dapat membatalkan penggunaan tingkatan data ketika memilih jenis simpul dari keluarga r6gd. Jika Anda mengatur parameter `--no-data-tiering`, operasi akan gagal.

Untuk Linux, macOS, atau Unix:

```
aws memorydb create-cluster \
```

```
--cluster-name my-cluster \  
--node-type db.r6gd.xlarge \  
--acl-name my-acl \  
--subnet-group my-sg \  
--data-tiering
```

Untuk Windows:

```
aws memorydb create-cluster ^  
  --cluster-name my-cluster ^  
  --node-type db.r6gd.xlarge ^  
  --acl-name my-acl ^  
  --subnet-group my-sg  
  --data-tiering
```

Setelah menjalankan operasi ini, Anda akan melihat respons seperti yang berikut ini:

```
{  
  "Cluster": {  
    "Name": "my-cluster",  
    "Status": "creating",  
    "NumberOfShards": 1,  
    "AvailabilityMode": "MultiAZ",  
    "ClusterEndpoint": {  
      "Port": 6379  
    },  
    "NodeType": "db.r6gd.xlarge",  
    "EngineVersion": "6.2",  
    "EnginePatchVersion": "6.2.6",  
    "ParameterGroupName": "default.memorydb-redis6",  
    "ParameterGroupStatus": "in-sync",  
    "SubnetGroupName": "my-sg",  
    "TLSEnabled": true,  
    "ARN": "arn:aws:memorydb:us-east-1:xxxxxxxxxxxx:cluster/my-cluster",  
    "SnapshotRetentionLimit": 0,  
    "MaintenanceWindow": "wed:03:00-wed:04:00",  
    "SnapshotWindow": "04:30-05:30",  
    "ACLName": "my-acl",  
    "DataTiering": "true",  
    "AutoMinorVersionUpgrade": true  
  }  
}
```

Memulihkan data dari snapshot ke cluster dengan tiering data diaktifkan

Anda dapat memulihkan snapshot ke cluster baru dengan tiering data yang diaktifkan menggunakan (Console), (AWS CLI) atau (MemoryDB API). Ketika Anda membuat sebuah klaster menggunakan jenis simpul dalam keluarga r6gd, tingkatan data diaktifkan.

Memulihkan data dari snapshot ke cluster dengan tiering data diaktifkan (konsol)

Untuk memulihkan snapshot ke cluster baru dengan tiering data diaktifkan (konsol), ikuti langkah-langkah di [Memulihkan dari snapshot \(Konsol\)](#)

Perhatikan bahwa untuk mengaktifkan data-tiering, Anda perlu memilih jenis node dari keluarga r6gd.

Memulihkan data dari snapshot ke cluster dengan tiering data diaktifkan (CLI)AWS

Saat membuat cluster menggunakan AWS CLI, tiering data secara default digunakan dengan memilih tipe node dari keluarga r6gd, seperti db.r6gd.xlarge dan mengatur parameter. `--data-tiering`

Anda tidak dapat membatalkan penggunaan tingkatan data ketika memilih jenis simpul dari keluarga r6gd. Jika Anda mengatur parameter `--no-data-tiering`, operasi akan gagal.

Untuk Linux, macOS, atau Unix:

```
aws memorydb create-cluster \  
  --cluster-name my-cluster \  
  --node-type db.r6gd.xlarge \  
  --acl-name my-acl \  
  --subnet-group my-sg \  
  --data-tiering \  
  --snapshot-name my-snapshot
```

Untuk Linux, macOS, atau Unix:

```
aws memorydb create-cluster ^  
  --cluster-name my-cluster ^  
  --node-type db.r6gd.xlarge ^  
  --acl-name my-acl ^  
  --subnet-group my-sg ^  
  --data-tiering ^  
  --snapshot-name my-snapshot
```

Setelah menjalankan operasi ini, Anda akan melihat respons seperti yang berikut ini:

```
{
  "Cluster": {
    "Name": "my-cluster",
    "Status": "creating",
    "NumberOfShards": 1,
    "AvailabilityMode": "MultiAZ",
    "ClusterEndpoint": {
      "Port": 6379
    },
    "NodeType": "db.r6gd.xlarge",
    "EngineVersion": "6.2",
    "EnginePatchVersion": "6.2.6",
    "ParameterGroupName": "default.memorydb-redis6",
    "ParameterGroupStatus": "in-sync",
    "SubnetGroupName": "my-sg",
    "TLSEnabled": true,
    "ARN": "arn:aws:memorydb:us-east-1:xxxxxxxxxxxx:cluster/my-cluster",
    "SnapshotRetentionLimit": 0,
    "MaintenanceWindow": "wed:03:00-wed:04:00",
    "SnapshotWindow": "04:30-05:30",
    "ACLName": "my-acl",
    "DataTiering": "true"
  }
}
```

Menyiapkan klaster

Berikut ini, Anda dapat menemukan petunjuk tentang membuat cluster menggunakan konsol MemoryDB, AWS CLI, atau MemoryDB API.

Setiap kali Anda membuat cluster, itu adalah ide yang baik untuk melakukan beberapa pekerjaan persiapan sehingga Anda tidak perlu meng-upgrade atau membuat perubahan segera.

Topik

- [Menentukan kebutuhan Anda](#)

Menentukan kebutuhan Anda

Persiapan

Mengetahui jawaban atas pertanyaan berikut akan membantu memudahkan proses pembuatan klaster Anda:

- Pastikan untuk membuat grup subnet di VPC yang sama sebelum Anda mulai membuat cluster. Atau, Anda dapat menggunakan grup subnet default yang disediakan. Untuk informasi selengkapnya, lihat [Subnet dan grup subnet](#).

MemoryDB dirancang untuk diakses dari dalam AWS menggunakan Amazon EC2. Namun, jika Anda meluncurkan dalam VPC berbasis Amazon VPC, Anda dapat memberikan akses dari luar. AWS Untuk informasi selengkapnya, lihat [Mengakses sumber daya MemoryDB dari luar AWS](#).

- Apakah Anda perlu menyesuaikan nilai parameter tertentu?

Jika ya, buat grup parameter kustom. Untuk informasi selengkapnya, lihat [Membuat grup parameter](#).

- Apakah Anda perlu membuat grup keamanan VPC?

Untuk informasi selengkapnya, lihat [Keamanan di VPC Anda](#).

- Bagaimana Anda akan menerapkan toleransi kesalahan?

Untuk informasi selengkapnya, lihat [Mitigasi Kegagalan](#).

Topik

- [Persyaratan memori dan prosesor](#)
- [Konfigurasi cluster MemoryDB](#)
- [Peningkatan I/O Multiplexing](#)
- [Persyaratan penskalaan](#)
- [Persyaratan akses](#)
- [Wilayah dan Zona Ketersediaan](#)

Persyaratan memori dan prosesor

Blok bangunan dasar MemoryDB adalah node. Node dikonfigurasi dalam pecahan untuk membentuk cluster. Saat menentukan jenis simpul yang akan digunakan untuk klaster Anda, pertimbangkan konfigurasi simpul klaster dan jumlah data yang harus disimpan.

Konfigurasi cluster MemoryDB

Cluster MemoryDB terdiri dari 1 hingga 500 pecahan. Data dalam cluster MemoryDB dipartisi di seluruh pecahan di cluster. Aplikasi Anda terhubung dengan cluster MemoryDB menggunakan alamat jaringan yang disebut Endpoint. Selain titik akhir node, cluster MemoryDB sendiri memiliki titik akhir yang disebut titik akhir cluster. Aplikasi Anda dapat menggunakan endpoint ini untuk membaca dari atau menulis ke cluster, meninggalkan penentuan node mana yang akan dibaca atau ditulis hingga MemoryDB.

Peningkatan I/O Multiplexing

Jika Anda menjalankan Redis OSS versi 7.0 atau lebih tinggi, Anda akan mendapatkan akselerasi tambahan dengan multiplexing I/O yang disempurnakan, di mana setiap jaringan pipa thread IO khusus memerintahkan dari beberapa klien ke mesin Redis OSS, memanfaatkan kemampuan Redis OSS untuk memproses perintah secara efisien dalam batch. Untuk informasi selengkapnya, lihat [Performa ultra-cepat](#) dan [the section called “Jenis simpul yang didukung”](#).

Persyaratan penskalaan

Semua cluster dapat ditingkatkan tipe node yang lebih besar. Saat Anda meningkatkan kluster MemoryDB, Anda dapat melakukannya secara online sehingga cluster tetap tersedia atau Anda dapat menyemai cluster baru dari snapshot dan menghindari cluster baru mulai kosong.

Untuk informasi selengkapnya, lihat [Penskalaan](#) dalam panduan ini.

Persyaratan akses

Secara desain, cluster MemoryDB diakses dari instans Amazon EC2. Akses jaringan ke cluster MemoryDB terbatas pada akun yang membuat cluster. Oleh karena itu, sebelum Anda dapat mengakses cluster dari instans Amazon EC2, Anda harus mengotorisasi ingress ke cluster. Untuk petunjuk yang lebih mendetail, lihat [Langkah 2: Otorisasi akses ke cluster](#) dalam panduan ini.

Wilayah dan Zona Ketersediaan

Dengan menemukan cluster MemoryDB Anda di AWS Wilayah yang dekat dengan aplikasi Anda, Anda dapat mengurangi latensi. Jika klaster Anda memiliki beberapa node, menempatkan node Anda di Availability Zone yang berbeda dapat mengurangi dampak kegagalan pada klaster Anda.

Untuk informasi selengkapnya, lihat berikut ini:

- [Memilih Wilayah dan Availability Zone](#)
- [Mitigasi Kegagalan](#)

Membuat klaster

MemoryDB menawarkan tiga cara untuk membuat cluster. Untuk informasi selengkapnya, lihat [Langkah 1: Buat cluster](#).

Melihat detail klaster

Anda dapat melihat informasi detail tentang satu atau beberapa cluster menggunakan konsol MemoryDB, AWS CLI, atau MemoryDB API.

Melihat detail untuk cluster MemoryDB (Konsol)

Prosedur berikut merinci cara melihat detail cluster MemoryDB menggunakan konsol MemoryDB.

1. [Masuk ke AWS Management Console dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/.](https://console.aws.amazon.com/memorydb/)
2. Untuk melihat detail cluster, pilih tombol radio di sebelah kiri nama cluster dan kemudian pilih Lihat detail. Anda juga dapat mengklik langsung pada cluster untuk melihat halaman detail cluster.

Halaman detail Cluster menampilkan detail tentang cluster, termasuk titik akhir cluster. Anda dapat melihat detail lebih lanjut menggunakan beberapa tab yang tersedia di halaman detail Cluster.

3. Pilih tab Pecahan dan node untuk melihat daftar pecahan cluster dan jumlah node di setiap pecahan.
4. Untuk melihat informasi spesifik pada sebuah node, perluas pecahan pada tabel di bawah ini. Atau Anda juga dapat mencari pecahan menggunakan kotak pencarian.

Melakukan hal ini menampilkan informasi tentang setiap node, termasuk Availability Zone, slot/keyspaces dan statusnya.

5. Pilih tab Metrik untuk memantau prosesnya masing-masing, seperti Pemanfaatan CPU dan Pemanfaatan CPU Mesin. Untuk informasi selengkapnya, lihat [Metrik untuk MemoryDB](#).
6. Pilih tab Jaringan dan keamanan untuk melihat detail grup subnet dan grup keamanan.
 - a. Di grup Subnet, Anda dapat melihat nama grup subnet, tautan ke VPC yang dimiliki subnet dan Nama Sumber Daya Amazon (ARN) grup subnet.
 - b. Di Grup keamanan, Anda dapat melihat ID grup keamanan, nama, dan deskripsi.
7. Pilih tab Maintenace dan snapshot untuk melihat detail pengaturan snapshot.
 - a. Di Snapshot, Anda dapat melihat apakah Snapshot Otomatis diaktifkan, periode retensi snapshot, dan jendela snapshot.

- b. Di Snapshots, Anda akan melihat daftar snapshot apa pun ke cluster ini, termasuk nama snapshot, ukuran, jumlah pecahan, dan status.

Untuk informasi selengkapnya, lihat [Melakukan snapshot dan pemulihan](#) .

8. Pilih tab Maintenance dan snapshot untuk melihat detail Jendela Pemeliharaan, bersama dengan pembaruan ACL, Resharding, atau Layanan yang tertunda. Untuk informasi selengkapnya, lihat [Mengelola pemeliharaan](#).
9. Pilih tab Pembaruan Layanan untuk melihat detail pembaruan layanan apa pun yang berlaku untuk klaster ini. Untuk informasi selengkapnya, lihat [Pembaruan layanan di MemoryDB](#).
10. Pilih tab Tag untuk melihat detail tag sumber daya atau alokasi biaya apa pun yang terkait dengan klaster ini. Untuk informasi selengkapnya, lihat [Menandai snapshot](#).

Melihat detail cluster (AWS CLI)

Anda dapat melihat detail untuk cluster menggunakan AWS CLI `describe-clusters` perintah. Jika parameter `--cluster-name` dihilangkan, detail untuk maksimal `--max-results` klaster akan ditampilkan. Jika parameter `--cluster-name` disertakan, detail untuk klaster yang ditentukan akan ditampilkan. Anda dapat membatasi jumlah catatan yang ditampilkan dengan parameter `--max-results`.

Kode berikut menampilkan daftar detail untuk `my-cluster`.

```
aws memorydb describe-clusters --cluster-name my-cluster
```

Kode berikut menampilkan daftar detail untuk maksimal 25 klaster.

```
aws memorydb describe-clusters --max-results 25
```

Example

Untuk Linux, macOS, atau Unix:

```
aws memorydb describe-clusters \  
  --cluster-name my-cluster \  
  --show-shard-details
```

Untuk Windows:

```
aws memorydb describe-clusters ^
  --cluster-name my-cluster ^
  --show-shard-details
```

Output JSON berikut menunjukkan respons:

```
{
  "Clusters": [
    {
      "Name": "my-cluster",
      "Description": "my cluster",
      "Status": "available",
      "NumberOfShards": 1,
      "Shards": [
        {
          "Name": "0001",
          "Status": "available",
          "Slots": "0-16383",
          "Nodes": [
            {
              "Name": "my-cluster-0001-001",
              "Status": "available",
              "AvailabilityZone": "us-east-1a",
              "CreateTime": 1629230643.961,
              "Endpoint": {
                "Address": "my-cluster-0001-001.my-
cluster.abcdef.memorydb.us-east-1.amazonaws.com",
                "Port": 6379
              }
            },
            {
              "Name": "my-cluster-0001-002",
              "Status": "available",
              "CreateTime": 1629230644.025,
              "Endpoint": {
                "Address": "my-cluster-0001-002.my-
cluster.abcdef.memorydb.us-east-1.amazonaws.com",
                "Port": 6379
              }
            }
          ],
          "NumberOfNodes": 2
        }
      ]
    }
  ]
}
```

```
    ],
    "ClusterEndpoint": {
      "Address": "clustercfg.my-cluster.abcdef.memorydb.us-
east-1.amazonaws.com",
      "Port": 6379
    },
    "NodeType": "db.r6g.large",
    "EngineVersion": "6.2",
    "EnginePatchVersion": "6.2.6",
    "ParameterGroupName": "default.memorydb-redis6",
    "ParameterGroupStatus": "in-sync",
    "SubnetGroupName": "default",
    "TLSEnabled": true,
    "ARN": "arn:aws:memorydb:us-east-1:0000000000:cluster/my-cluster",
    "SnapshotRetentionLimit": 0,
    "MaintenanceWindow": "sat:06:30-sat:07:30",
    "SnapshotWindow": "04:00-05:00",
    "ACLName": "open-access",
    "DataTiering": "false",
    "AutoMinorVersionUpgrade": true,
  }
}
```

Untuk informasi lebih lanjut, lihat topik AWS CLI untuk MemoryDB. [describe-clusters](#)

Melihat detail cluster (MemoryDB API)

Anda dapat melihat detail untuk klaster menggunakan aksi API DescribeClusters MemoryDB. Jika parameter `ClusterName` disertakan, detail untuk klaster yang ditentukan akan ditampilkan. Jika parameter `ClusterName` dihilangkan, detail untuk maksimal `MaxResults` klaster (default-nya 100) akan ditampilkan. Nilai untuk `MaxResults` tidak boleh kurang dari 20 atau lebih dari 100.

Kode berikut menampilkan daftar detail untuk `my-cluster`.

```
https://memory-db.us-east-1.amazonaws.com/
?Action=DescribeClusters
&ClusterName=my-cluster
&Version=2021-01-01
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20210802T192317Z
&X-Amz-Credential=<credential>
```


Kode berikut menampilkan daftar detail untuk maksimal 25 klaster.

```
https://memory-db.us-east-1.amazonaws.com/  
?Action=DescribeClusters  
&MaxResults=25  
&Version=2021-02-02  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20210802T192317Z  
&X-Amz-Credential=<credential>
```

Untuk informasi selengkapnya, lihat topik referensi API MemoryDB. [DescribeClusters](#)

Memodifikasi cluster MemoryDB

Selain menambahkan atau menghapus node dari cluster, mungkin ada saat-saat di mana Anda perlu membuat perubahan lain ke cluster yang ada, seperti menambahkan grup keamanan, mengubah jendela pemeliharaan atau grup parameter.

Sebaiknya atur periode pemeliharaan Anda pada waktu penggunaan terendah. Jadi, Anda mungkin perlu mengubahnya dari waktu ke waktu.

Saat Anda mengubah parameter klaster, perubahan tersebut akan segera diterapkan pada klaster tersebut. Hal ini berlaku terlepas dari apakah Anda mengubah grup parameter klaster itu sendiri atau nilai parameter dalam grup parameter klaster.

Anda juga dapat memperbarui versi mesin cluster Anda. Misalnya, Anda dapat memilih versi minor engine baru dan MemoryDB akan segera memperbarui cluster Anda.

Menggunakan AWS Management Console

Untuk mengubah klaster

1. [Masuk ke AWS Management Console dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/.](https://console.aws.amazon.com/memorydb/)
2. Dari daftar di sudut kanan atas, pilih AWS Wilayah tempat cluster yang ingin Anda modifikasi berada.
3. Dari navigasi kiri, pergi ke Clusters. Dari detail Clusters, pilih cluster menggunakan tombol radio dan pergi ke Actions dan kemudian Modify.
4. Halaman Modify muncul.
5. Di jendela Modify, buat modifikasi yang Anda inginkan. Opsinya meliputi:
 - Deskripsi
 - Grup subnet
 - Grup Keamanan VPC
 - Jenis simpul

Note

Jika klaster menggunakan jenis simpul dari keluarga r6gd, Anda hanya dapat memilih ukuran simpul yang berbeda dari dalam keluarga tersebut. Jika Anda memilih jenis

simpul dari keluarga r6gd, tingkatan data akan diaktifkan secara otomatis. Untuk informasi selengkapnya, lihat [Tingkatan data](#).

- Kompatibilitas versi Redis OSS
- Aktifkan snapshot otomatis
- Periode Retensi Snapshot
- Jendela Snapshot
- Periode pemeliharaan
- Topik untuk Notifikasi SNS

6. Pilih Simpan perubahan.

Anda juga dapat pergi ke halaman detail Cluster dan klik modifikasi untuk membuat modifikasi pada cluster. Jika Anda ingin memodifikasi bagian tertentu dari cluster, Anda dapat pergi ke tab masing-masing di halaman detail Cluster dan klik Ubah.

Menggunakan AWS CLI

Anda dapat memodifikasi cluster yang ada menggunakan AWS CLI `update-cluster` operasi. Untuk mengubah nilai konfigurasi klaster, tentukan ID klaster, parameter yang akan diubah, dan nilai baru parameter. Contoh berikut mengubah periode pemeliharaan untuk klaster bernama `my-cluster` dan menerapkan perubahan tersebut secara langsung.

Untuk Linux, macOS, atau Unix:

```
aws memorydb update-cluster \  
  --cluster-name my-cluster \  
  --preferred-maintenance-window sun:23:00-mon:02:00
```

Untuk Windows:

```
aws memorydb update-cluster ^  
  --cluster-name my-cluster ^  
  --preferred-maintenance-window sun:23:00-mon:02:00
```

Untuk informasi selengkapnya, lihat [update-cluster](#) di Command Reference. AWS CLI

Menggunakan MemoryDB API

Anda dapat memodifikasi cluster yang ada menggunakan operasi API [UpdateClusterMemoryDB](#). Untuk mengubah nilai konfigurasi klaster, tentukan ID klaster, parameter yang akan diubah, dan nilai baru parameter. Contoh berikut mengubah periode pemeliharaan untuk klaster bernama `my-cluster` dan menerapkan perubahan tersebut secara langsung.

```
https://memory-db.us-east-1.amazonaws.com/  
?Action=UpdateCluster  
&ClusterName=my-cluster  
&PreferredMaintenanceWindow=sun:23:00-mon:02:00  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20210801T220302Z  
&X-Amz-Algorithm=Amazon4-HMAC-SHA256  
&X-Amz-Date=20210802T220302Z  
&X-Amz-SignedHeaders=Host  
&X-Amz-Expires=20210801T220302Z  
&X-Amz-Credential=<credential>  
&X-Amz-Signature=<signature>
```

Menambahkan/Menghapus node dari cluster

Anda dapat menambahkan atau menghapus node dari cluster menggunakan API AWS Management Console, the AWS CLI, atau MemoryDB.

Menggunakan AWS Management Console

1. [Masuk ke AWS Management Console dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/.](https://console.aws.amazon.com/memorydb/)
2. Dari daftar cluster, pilih nama cluster dari mana Anda ingin menambahkan atau menghapus node.
3. Di bawah tab Pecahan dan node, pilih Tambah/Hapus node
4. Di Jumlah node baru, masukkan jumlah node yang Anda inginkan.
5. Pilih Konfirmasi.

Important

Jika Anda mengatur jumlah node ke 1, Anda tidak akan lagi diaktifkan Multi-AZ. Anda juga dapat memilih untuk mengaktifkan Auto failover.

Menggunakan AWS CLI

1. Identifikasi nama-nama node yang ingin Anda hapus. Untuk informasi selengkapnya, lihat [Melihat detail klaster.](#)
2. Gunakan operasi CLI `update-cluster` dengan daftar simpul yang akan dihapus, seperti dalam contoh berikut.

Untuk menghapus simpul dari klaster menggunakan antarmuka baris perintah, gunakan perintah `update-cluster` dengan parameter berikut:

- `--cluster-nameID` dari cluster yang ingin Anda hapus node dari.
- `--replica-configuration`— Memungkinkan Anda untuk mengatur jumlah replika:
 - `ReplicaCount`— Tetapkan properti ini untuk menentukan jumlah node replika yang Anda inginkan.
- `--region` Menentukan AWS Wilayah cluster yang ingin Anda hapus node dari.

Untuk Linux, macOS, atau Unix:

```
aws memorydb update-cluster \  
  --cluster-name my-cluster \  
  --replica-configuration \  
    ReplicaCount=1 \  
  --region us-east-1
```

Untuk Windows:

```
aws memorydb update-cluster ^  
  --cluster-name my-cluster ^  
  --replica-configuration ^  
    ReplicaCount=1 ^  
  --region us-east-1
```

Untuk informasi lebih lanjut, lihat AWS CLI topiknya [update-cluster](#).

Menggunakan MemoryDB API

Untuk menghapus node menggunakan API MemoryDB, panggil operasi UpdateCluster API dengan nama cluster dan daftar node yang akan dihapus, seperti yang ditunjukkan:

- `ClusterNameID` dari cluster yang ingin Anda hapus node dari.
- `ReplicaConfiguration`— Memungkinkan Anda untuk mengatur jumlah replika:
 - `ReplicaCount`— Tetapkan properti ini untuk menentukan jumlah node replika yang Anda inginkan.
- `Region` Menentukan AWS Wilayah cluster yang Anda ingin menghapus node dari.

Untuk informasi lebih lanjut, lihat [UpdateCluster](#).

Mengakses klaster Anda

Instans MemoryDB Anda dirancang untuk diakses melalui instans Amazon EC2.

Anda dapat mengakses node MemoryDB Anda dari instans Amazon EC2 di VPC Amazon yang sama. Atau, dengan menggunakan VPC peering, Anda dapat mengakses node MemoryDB Anda dari Amazon EC2 di VPC Amazon yang berbeda.

Topik

- [Berikan akses ke cluster Anda](#)
- [Mengakses sumber daya MemoryDB dari luar AWS](#)

Berikan akses ke cluster Anda

Anda dapat terhubung ke cluster MemoryDB Anda hanya dari instans Amazon EC2 yang berjalan di VPC Amazon yang sama. Dalam hal ini, Anda akan perlu memberikan izin masuk jaringan ke klaster.

Untuk memberikan izin masuk jaringan dari grup keamanan Amazon VPC ke klaster

1. [Masuk ke AWS Management Console dan buka konsol Amazon EC2 di https://console.aws.amazon.com/ec2/.](https://console.aws.amazon.com/ec2/)
2. Di panel navigasi kiri, di bawah Jaringan & Keamanan, pilih Grup Keamanan.
3. Dari daftar grup keamanan, pilih grup keamanan untuk Amazon VPC Anda. Kecuali Anda membuat grup keamanan untuk penggunaan MemoryDB, grup keamanan ini akan diberi nama default.
4. Pilih tab Masuk, lalu lakukan hal berikut:
 - a. Pilih Edit.
 - b. Pilih Tambahkan aturan.
 - c. Di kolom Jenis, pilih Aturan TCP kustom.
 - d. Di kotak Rentang port, ketik nomor port untuk simpul klaster Anda. Nomor ini harus sama dengan yang Anda tentukan saat meluncurkan klaster. Port default untuk Redis OSS adalah. **6379**
 - e. Di kotak Sumber, pilih Di mana saja yang memiliki rentang port (0.0.0.0/0) sehingga instans Amazon EC2 apa pun yang Anda luncurkan dalam VPC Amazon Anda dapat terhubung ke node MemoryDB Anda.

⚠ Important

Membuka cluster MemoryDB ke 0.0.0.0/0 tidak mengekspos cluster ke Internet karena tidak memiliki alamat IP publik dan oleh karena itu tidak dapat diakses dari luar VPC. Namun, grup keamanan default dapat diterapkan ke instans Amazon EC2 lainnya di akun pelanggan, dan instans tersebut mungkin memiliki alamat IP publik. Jika instans tersebut menjalankan sesuatu di port default, layanan tersebut dapat terekspos secara tak disengaja. Oleh karena itu, kami sarankan untuk membuat Grup Keamanan VPC yang akan digunakan secara eksklusif oleh MemoryDB. Untuk informasi selengkapnya, lihat [Grup Keamanan Kustom](#).

- f. Pilih Simpan.

Saat meluncurkan instans Amazon EC2 ke VPC Amazon Anda, instans tersebut akan dapat terhubung ke cluster MemoryDB Anda.

Mengakses sumber daya MemoryDB dari luar AWS

MemoryDB adalah layanan yang dirancang untuk digunakan secara internal ke VPC Anda. Akses eksternal tidak disarankan karena latensi lalu lintas Internet dan masalah keamanan. Namun, jika akses eksternal ke MemoryDB diperlukan untuk tujuan pengujian atau pengembangan, itu dapat dilakukan melalui VPN.

Menggunakan AWS Client VPN, Anda mengizinkan akses eksternal ke node MemoryDB Anda dengan manfaat berikut:

- Akses terbatas hanya untuk pengguna atau kunci autentikasi yang disetujui;
- Lalu lintas terenkripsi antara Klien VPN dan titik akhir AWS VPN;
- Akses yang terbatas ke subnet atau simpul tertentu;
- Pencabutan akses dengan mudah dari pengguna atau kunci autentikasi;
- Mengaudit koneksi;

Prosedur berikut menunjukkan cara untuk:

Topik

- [Membuat otoritas sertifikat](#)
- [Mengkonfigurasi komponen VPN AWS klien](#)
- [Konfigurasi klien VPN](#)

Membuat otoritas sertifikat

Dimungkinkan untuk membuat Otoritas Sertifikat (CA) menggunakan teknik atau alat yang berbeda. Dianjurkan menggunakan utilitas `easy-rsa`, yang disediakan oleh proyek [OpenVPN](#). Terlepas opsi pilihan Anda, pastikan untuk menjaga kunci tetap aman. Prosedur berikut mengunduh skrip `easy-rsa`, membuat Otoritas Sertifikat dan kunci untuk autentikasi klien VPN pertama:

- Untuk membuat sertifikat awal, buka terminal dan lakukan hal berikut:
 - `git clone https://github.com/OpenVPN/easy-rsa`
 - `cd easy-rsa`
 - `./easyrsa3/easyrsa init-pki`
 - `./easyrsa3/easyrsa build-ca nopass`

- `./easyrsa3/easyrsa build-server-full server nopass`
- `./easyrsa3/easyrsa build-client-full client1.domain.tld nopass`

Subdirektori pki yang berisi sertifikat akan dibuat di bawah easy-rsa.

- Kirim sertifikat server ke manajer AWS Sertifikat (ACM):
 - Pada konsol ACM, pilih Certificate Manager.
 - Pilih Impor Sertifikat.
 - Masukkan sertifikat kunci publik yang tersedia di dalam file `easy-rsa/pki/issued/server.crt` pada bidang Tubuh sertifikat.
 - Tempelkan kunci privat yang tersedia di dalam `easy-rsa/pki/private/server.key` pada bidang Kunci privat sertifikat. Pastikan untuk mengblok semua baris di antara `BEGIN AND END PRIVATE KEY` (termasuk baris `BEGIN` dan `END`).
 - Tempelkan kunci publik CA yang tersedia pada file `easy-rsa/pki/ca.crt` pada bidang Rantai sertifikat.
 - Pilih Tinjau dan impor.
 - Pilih Impor.

Untuk mengirimkan sertifikat server ke ACM menggunakan AWS CLI, jalankan perintah berikut:

```
aws acm import-certificate --certificate file://easy-rsa/pki/issued/server.crt --private-key file://easy-rsa/pki/private/server.key --certificate-chain file://easy-rsa/pki/ca.crt --region region
```

Perhatikan ARN sertifikat untuk penggunaan di masa mendatang.

Mengkonfigurasi komponen VPN AWS klien

Menggunakan AWS Konsol

Di AWS konsol, pilih Layanan dan kemudian VPC.

Di bawah Virtual Private Network, pilih Titik akhir Client VPN dan lakukan hal berikut:

Mengkonfigurasi komponen AWS Client VPN

- Pilih Buat Titik Akhir Client VPN.
- Tentukan opsi berikut:

Mengakses MemoryDB dari luar AWS

- CIDR IPv4 Klien: gunakan jaringan privat dengan netmask setidaknya pada rentang /22. Pastikan bahwa subnet yang dipilih tidak bertentangan dengan alamat dari jaringan VPC. Contoh: 10.0.0.0/22.
- Pada ARN sertifikat server, pilih ARN dari sertifikat yang sebelumnya diimpor.
- Pilih Gunakan autentikasi bersama.
- Pada ARN sertifikat klien, pilih ARN dari sertifikat yang sebelumnya diimpor.
- Pilih Buat Titik Akhir Client VPN.

Menggunakan AWS CLI

Jalankan perintah berikut:

```
aws ec2 create-client-vpn-endpoint --client-cidr-block
"10.0.0.0/22" --server-certificate-arn arn:aws:acm:us-
east-1:012345678912:certificate/0123abcd-ab12-01a0-123a-123456abcdef --
authentication-options Type=certificate-
authentication, ,MutualAuthentication={ClientRootCertificateChainArn=arn:aws:acm:
east-1:012345678912:certificate/123abcd-ab12-01a0-123a-123456abcdef} --
connection-log-options Enabled=false
```

Contoh keluaran:

```
"ClientVpnEndpointId": "cvpn-endpoint-0123456789abcdefg",
"Status": { "Code": "pending-associate" }, "DnsName": "cvpn-
endpoint-0123456789abcdefg.prod.clientvpn.us-east-1.amazonaws.com" }
```

Kaitkan jaringan target ke titik akhir VPN

- Pilih titik akhir baru VPN, dan kemudian pilih tab Asosiasi.
- Pilih Associate dan tentukan opsi berikut.
 - VPC: Pilih VPC MemoryDB Cluster.
 - Pilih salah satu jaringan cluster MemoryDB. Jika ragu, tinjau jaringan di Grup Subnet di dasbor MemoryDB.
 - Pilih Associate. Jika perlu, ulangi langkah tersebut untuk jaringan yang tersisa.

Menggunakan AWS CLI

Jalankan perintah berikut:

```
aws ec2 associate-client-vpn-target-network --client-vpn-endpoint-id cvpn-  
endpoint-0123456789abcdefg --subnet-id subnet-0123456789abcdef
```

Contoh keluaran:

```
"Status": { "Code": "associating" }, "AssociationId": "cvpn-  
assoc-0123456789abcdef" }
```

Tinjau grup keamanan VPN

Titik akhir VPN akan secara otomatis mengadopsi grup keamanan default dari VPC. Periksa aturan masuk dan keluar dan konfirmasikan apakah grup keamanan mengizinkan lalu lintas dari jaringan VPN (ditentukan pada pengaturan Titik Akhir VPN) ke jaringan MemoryDB pada port layanan (secara default, 6379 untuk Redis).

Jika Anda perlu mengubah grup keamanan yang ditetapkan untuk titik akhir VPN, lanjutkan sebagai berikut:

- Pilih grup keamanan saat ini.
- Pilih Terapkan Grup Keamanan.
- Pilih Grup Keamanan baru.

Menggunakan AWS CLI

Jalankan perintah berikut:

```
aws ec2 apply-security-groups-to-client-vpn-target-network --  
client-vpn-endpoint-id cvpn-endpoint-0123456789abcdefga --vpc-id  
vpc-0123456789abcdef --security-group-ids sg-0123456789abcdef
```

Contoh keluaran:

```
"SecurityGroupIds": [ "sg-0123456789abcdef" ] }
```

Note

Grup keamanan MemoryDB juga perlu mengizinkan lalu lintas yang datang dari klien VPN. Alamat klien akan ditutupi dengan alamat titik akhir VPN, sesuai Jaringan VPC. Oleh karena

itu, pertimbangkan jaringan VPC (bukan jaringan Klien VPN) saat membuat aturan masuk pada grup keamanan MemoryDB.

Otorisasi akses VPN ke jaringan tujuan

Pada tab Otorisasi, pilih Izinkan Masuk dan tentukan hal berikut:

- Jaringan tujuan untuk mengaktifkan akses: Gunakan 0.0.0.0/0 untuk memungkinkan akses ke jaringan apa pun (termasuk Internet) atau membatasi jaringan/host MemoryDB.
- Di bawah Berikan akses ke:, pilih Izinkan akses ke semua pengguna.
- Pilih Tambahkan Aturan Otorisasi.

Menggunakan AWS CLI

Jalankan perintah berikut:

```
aws ec2 authorize-client-vpn-ingress --client-vpn-endpoint-id cvpn-  
endpoint-0123456789abcdefg --target-network-cidr 0.0.0.0/0 --authorize-all-  
groups
```

Contoh keluaran:

```
{ "Status": { "Code": "authorizing" } }
```

Mengizinkan akses ke Internet dari klien VPN

Jika Anda perlu menelusuri Internet melalui VPN, Anda perlu membuat rute tambahan. Pilih tab Tabel Rute dan kemudian pilih Buat Rute:

- Tujuan rute: 0.0.0.0/0
- ID Subnet VPC Target: Pilih salah satu subnet yang terkait dengan akses ke Internet.
- Pilih Buat Rute.

Menggunakan AWS CLI

Jalankan perintah berikut:

```
aws ec2 create-client-vpn-route --client-vpn-endpoint-id cvpn-  
endpoint-0123456789abcdefg --destination-cidr-block 0.0.0.0/0 --target-vpc-  
subnet-id subnet-0123456789abcdef
```

Contoh keluaran:

```
{ "Status": { "Code": "creating" } }
```

Konfigurasi klien VPN

Di Dasbor AWS Client VPN, pilih titik akhir VPN yang baru saja dibuat dan pilih Unduh Konfigurasi Klien. Salin file konfigurasi, serta file `easy-rsa/pki/issued/client1.domain.tld.crt` dan `easy-rsa/pki/private/client1.domain.tld.key`. Edit file konfigurasi dan ubah atau tambahkan parameter berikut:

- `cert`: tambahkan baris baru dengan parameter `cert` menunjuk ke file `client1.domain.tld.crt`. Gunakan jalur lengkap ke file. Contoh: `cert /home/user/.cert/client1.domain.tld.crt`
- `cert: key`: tambahkan baris baru dengan kunci parameter menunjuk ke file `client1.domain.tld.key`. Gunakan jalur lengkap ke file. Contoh: `key /home/user/.cert/client1.domain.tld.key`

Tetapkan koneksi VPN dengan perintah: `sudo openvpn --config downloaded-client-config.ovpn`

Mencabut akses

Jika Anda perlu untuk membatalkan akses dari kunci klien tertentu, kunci tersebut perlu dicabut di CA. Kemudian kirimkan daftar pencabutan ke AWS Client VPN.

Mencabut kunci dengan `easy-rsa`:

- `cd easy-rsa`
- `./easyrsa3/easyrsa revoke client1.domain.tld`
- Masukkan “ya” untuk melanjutkan, atau masukkan input lain apa pun untuk membatalkan.

Continue with revocation: `yes` ... * `./easyrsa3/easyrsa gen-crl`
- CRL yang diperbarui telah dibuat. File CRL: `/home/user/easy-rsa/pki/crl.pem`

Mengimpor daftar pencabutan ke Client VPN AWS :

- Pada AWS Management Console, pilih Layanan dan kemudian VPC.
- Pilih Titik Akhir Client VPN.
- Pilih Titik Akhir Client VPN dan kemudian pilih Tindakan -> Impor CRL Sertifikat Klien.
- Tempelkan isi dari file `crl.pem`.

Menggunakan AWS CLI

Jalankan perintah berikut:

```
aws ec2 import-client-vpn-client-certificate-revocation-list --certificate-revocation-list file:///./easy-rsa/pki/crl.pem --client-vpn-endpoint-id cvpn-endpoint-0123456789abcdefg
```

Contoh keluaran:

```
Example output: { "Return": true }
```

Menemukan titik akhir koneksi

Aplikasi Anda terhubung ke cluster Anda menggunakan endpoint. Endpoint adalah alamat unik cluster. Gunakan Cluster Endpoint cluster untuk semua operasi.

Bagian berikut memandu Anda untuk menemukan titik akhir yang Anda perlukan.

Menemukan Endpoint untuk Cluster MemoryDB ()AWS Management Console

Untuk menemukan titik akhir cluster MemoryDB

1. [Masuk ke AWS Management Console dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/.](https://console.aws.amazon.com/memorydb/)
2. Dari panel navigasi, pilih Clusters.

Layar cluster akan muncul dengan daftar cluster. Pilih cluster yang ingin Anda sambungkan.

3. Untuk menemukan titik akhir cluster, pilih nama cluster (bukan tombol radio).
4. Titik akhir Cluster ditampilkan di bawah detail Cluster. Untuk menyalinnya, pilih ikon salin di sebelah kiri titik akhir.

Menemukan Titik Akhir untuk Cluster MemoryDB (CLI)AWS

Anda dapat menggunakan `describe-clusters` perintah untuk menemukan titik akhir untuk sebuah cluster. Perintah mengembalikan titik akhir cluster.

Operasi berikut mengambil titik akhir, yang dalam contoh ini direpresentasikan sebagai *sampel*, untuk cluster. `mycluster`

la mengembalikan respon JSON berikut:

```
aws memorydb describe-clusters \  
  --cluster-name mycluster
```

Untuk Windows:

```
aws memorydb describe-clusters ^  
  --cluster-name mycluster
```

```
{  
  "Clusters": [  
    {  
      "Name": "my-cluster",  
      "Status": "available",  
      "NumberOfShards": 1,  
      "ClusterEndpoint": {
```



```
        "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-  
east-1.amazonaws.com",  
        "Port": 6379  
    },  
    "NodeType": "db.r6g.large",  
    "EngineVersion": "6.2",  
    "EnginePatchVersion": "6.2.4",  
    "ParameterGroupName": "default.memorydb-redis6",  
    "ParameterGroupStatus": "in-sync",  
    "SubnetGroupName": "my-sg",  
    "TLSEnabled": true,  
    "ARN": "arn:aws:memorydb:us-east-1:zzzexamplearn:cluster/my-cluster",  
    "SnapshotRetentionLimit": 0,  
    "MaintenanceWindow": "wed:03:00-wed:04:00",  
    "SnapshotWindow": "04:30-05:30",  
    "ACLName": "my-acl",  
    "AutoMinorVersionUpgrade": true  
    }  
  ]  
}
```

Untuk informasi selengkapnya, lihat [menjelaskan klaster](#).

Menemukan Endpoint untuk Cluster MemoryDB (MemoryDB API)

Anda dapat menggunakan API MemoryDB untuk menemukan titik akhir cluster.

Menemukan Endpoint untuk Cluster MemoryDB (MemoryDB API)

Anda dapat menggunakan API MemoryDB untuk menemukan titik akhir kluster dengan tindakan tersebut. `DescribeClusters` Tindakan mengembalikan titik akhir cluster.

Operasi berikut mengambil endpoint cluster untuk cluster. `mycluster`

```
https://memory-db.us-east-1.amazonaws.com/  
?Action=DescribeClusters  
&ClusterName=mycluster  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20210802T192317Z  
&Version=2021-01-01  
&X-Amz-Credential=<credential>
```

Untuk informasi lebih lanjut, lihat [DescribeClusters](#).

Menggunakan serpihan

Shard adalah kumpulan dari satu hingga 6 node. Anda dapat membuat cluster dengan jumlah pecahan yang lebih tinggi dan jumlah replika yang lebih rendah dengan total hingga 500 node per cluster. Konfigurasi cluster ini dapat berkisar dari 500 pecahan dan 0 replika hingga 100 pecahan dan 4 replika, yang merupakan jumlah maksimum replika yang diizinkan. Data kluster dipartisi di seluruh serpihan kluster. Jika dalam serpihan terdapat lebih dari satu simpul, serpihan akan mengimplementasikan replikasi dengan satu simpul menjadi simpul primer baca/tulis dan simpul lain menjadi simpul replika hanya baca.

Saat Anda membuat cluster MemoryDB menggunakan AWS Management Console, Anda menentukan jumlah pecahan di cluster dan jumlah node dalam pecahan. Untuk informasi selengkapnya, lihat [Membuat cluster MemoryDB](#).

Setiap simpul dalam serpihan memiliki spesifikasi komputasi, penyimpanan, dan memori yang sama. MemoryDB API memungkinkan Anda mengontrol atribut seluruh cluster, seperti jumlah node, pengaturan keamanan, dan jendela pemeliharaan sistem.

Untuk informasi selengkapnya, lihat [Resharding offline dan penyeimbangan kembali shard untuk MemoryDB](#) dan [Resharding online dan penyeimbangan kembali shard untuk MemoryDB](#).

Menemukan nama pecahan

Anda dapat menemukan nama pecahan menggunakan API AWS Management Console, AWS CLI atau MemoryDB.

Menggunakan AWS Management Console

Prosedur berikut menggunakan AWS Management Console untuk menemukan nama pecahan cluster MemoryDB ini.

1. [Masuk ke AWS Management Console dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/](https://console.aws.amazon.com/memorydb/).
2. Di panel navigasi kiri, pilih Clusters.
3. Pilih cluster di bawah Nama yang nama pecahan yang ingin Anda temukan.
4. Di bawah tab Pecahan dan node, lihat daftar pecahan di bawah Nama. Anda juga dapat memperluas masing-masing untuk melihat detail node mereka.

Menggunakan AWS CLI

Untuk menemukan nama shard (shard) untuk cluster MemoryDB gunakan AWS CLI operasi `describe-clusters` dengan parameter opsional berikut.

- **--cluster-name**—Parameter opsional yang bila digunakan membatasi output ke rincian cluster yang ditentukan. Jika parameter ini dihilangkan, detail hingga 100 cluster dikembalikan.
- **--show-shard-details**—Mengembalikan detail pecahan, termasuk namanya.

Perintah ini akan menampilkan detail untuk `my-cluster`.

Untuk Linux, macOS, atau Unix:

```
aws memorydb describe-clusters \  
  --cluster-name my-cluster \  
  --show-shard-details
```

Untuk Windows:

```
aws memorydb describe-clusters ^  
  --cluster-name my-cluster  
  --show-shard-details
```

Ia mengembalikan respon JSON berikut:

Jeda baris ditambahkan agar dapat lebih mudah dibaca.

```
{  
  "Clusters": [  
    {  
      "Name": "my-cluster",  
      "Status": "available",  
      "NumberOfShards": 1,  
      "Shards": [  
        {  
          "Name": "0001",  
          "Status": "available",  
          "Slots": "0-16383",  
          "Nodes": [  
            {  
              "Name": "my-cluster-0001-001",  
              "Status": "available",  
              "AvailabilityZone": "us-east-1a",  
              "CreateTime": "2021-08-21T20:22:12.405000-07:00",  
              "Endpoint": {  
                "Address": "clustercfg.my-cluster.xxxxx.memorydb.us-  
east-1.amazonaws.com",  
                "Port": 6379  
              }  
            },  
            {  
              "Name": "my-cluster-0001-002",  
              "Status": "available",  
              "AvailabilityZone": "us-east-1b",  
              "CreateTime": "2021-08-21T20:22:12.405000-07:00",  
              "Endpoint": {  
                "Address": "clustercfg.my-cluster.xxxxx.memorydb.us-  
east-1.amazonaws.com",  
                "Port": 6379  
              }  
            }  
          ]  
        }  
      ]  
    }  
  ]  
}
```

```

        }
    }
    ],
    "NumberOfNodes": 2
}
],
"ClusterEndpoint": {
    "Address": "clustercfg.my-cluster.xxxxx.memorydb.us-east-1.amazonaws.com",
    "Port": 6379
},
"NodeType": "db.r6g.large",
"EngineVersion": "6.2",
"EnginePatchVersion": "6.2.6",
"ParameterGroupName": "default.memorydb-redis6",
"ParameterGroupStatus": "in-sync",
"SubnetGroupName": "my-sg",
"TLSEnabled": true,
"ARN": "arn:aws:memorydb:us-east-1:xxxxxexamplearn:cluster/my-cluster",
"SnapshotRetentionLimit": 0,
"MaintenanceWindow": "wed:03:00-wed:04:00",
"SnapshotWindow": "04:30-05:30",
"ACLName": "my-acl",
"DataTiering": "false",
"AutoMinorVersionUpgrade": true
}
]
}

```

Menggunakan MemoryDB API

Untuk menemukan id pecahan untuk cluster MemoryDB gunakan operasi API DescribeClusters dengan parameter opsional berikut.

- **ClusterName**—Parameter opsional yang bila digunakan membatasi output ke rincian cluster yang ditentukan. Jika parameter ini dihilangkan, detail hingga 100 cluster dikembalikan.
- **ShowShardDetails**—Mengembalikan detail pecahan, termasuk namanya.

Example

Perintah ini akan menampilkan detail untuk `my-cluster`.

Untuk Linux, macOS, atau Unix:

```
https://memory-db.us-east-1.amazonaws.com/  
?Action=DescribeClusters  
&ClusterName=sample-cluster  
&ShowShardDetails=true  
&Version=2021-01-01  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20210802T192317Z  
&X-Amz-Credential=<credential>
```

Mengelola implementasi MemoryDB Anda

Di bagian ini, Anda dapat menemukan detail tentang cara mengelola berbagai komponen implementasi MemoryDB Anda.

Topik

- [Versi mesin Redis OSS](#)
- [Mulai menggunakan JSON](#)
- [Menandai sumber daya MemoryDB Anda](#)
- [Mengelola pemeliharaan](#)
- [Praktik terbaik](#)
- [Memahami replikasi MemoryDB](#)
- [Melakukan snapshot dan pemulihan](#)
- [Penskalaan](#)
- [Mengonfigurasi parameter mesin menggunakan grup parameter](#)
- [Tutorial: Mengonfigurasi fungsi Lambda untuk mengakses MemoryDB di VPC Amazon](#)

Versi mesin Redis OSS

Bagian ini mencakup versi mesin Redis OSS yang didukung.

Topik

- [MemoryDB versi 7.1 \(ditingkatkan\)](#)
- [MemoryDB versi 7.0 \(ditingkatkan\)](#)
- [MemoryDB versi 6.2 \(ditingkatkan\)](#)
- [Meningkatkan versi mesin](#)

MemoryDB versi 7.1 (ditingkatkan)

MemoryDB versi 7.1 menambahkan dukungan untuk kemampuan pencarian vektor dalam pratinjau untuk wilayah tertentu, serta perbaikan bug penting dan peningkatan kinerja.

- **Fitur Pencarian Vektor:** Pencarian vektor dapat digunakan dengan fungsionalitas MemoryDB yang ada. Aplikasi yang tidak menggunakan pencarian vektor tidak akan terpengaruh oleh keberadaannya. Pratinjau pencarian vektor tersedia dalam MemoryDB versi 7.1 dan seterusnya di wilayah berikut: AS Timur (Virginia N. dan Ohio), AS Barat (Oregon), UE (Irlandia), dan Asia Pasifik (Tokyo). Silakan lihat dokumentasi di [sini](#) untuk cara mengaktifkan pratinjau pencarian vektor dan kemampuan fitur terkait.

Note

MemoryDB versi 7.1 kompatibel dengan Redis OSS v7.0. Untuk informasi lebih lanjut tentang rilis Redis OSS 7.0, lihat Catatan Rilis [Redis OSS 7.0 di Redis OSS](#) pada GitHub

MemoryDB versi 7.0 (ditingkatkan)

MemoryDB 7.0 menambahkan sejumlah perbaikan dan dukungan untuk fungsionalitas baru:

- **Fungsi Redis OSS:** MemoryDB 7 menambahkan dukungan untuk Fungsi Redis OSS, dan memberikan pengalaman terkelola yang memungkinkan pengembang untuk mengeksekusi [skrip LUA](#) dengan logika aplikasi yang disimpan di cluster MemoryDB, tanpa mengharuskan klien untuk mengirim ulang skrip ke server dengan setiap koneksi.
- **Perbaikan ACL:** MemoryDB 7 menambahkan dukungan untuk versi berikutnya dari Redis OSS Access Control Lists (ACL). Dengan MemoryDB OSS 7, klien sekarang dapat menentukan beberapa set izin pada kunci atau ruang kunci tertentu di Redis OSS.
- **Sharded Pub/Sub:** MemoryDB 7 menambahkan dukungan untuk menjalankan fungsionalitas Redis OSS Pub/Sub dengan cara sharded saat menjalankan MemoryDB di Cluster Mode Enabled (CME). Kemampuan Redis OSS Pub/Sub memungkinkan penerbit untuk mengeluarkan pesan ke sejumlah pelanggan di saluran. Dengan Amazon MemoryDB OSS 7, saluran terikat pada pecahan di cluster MemoryDB, menghilangkan kebutuhan untuk menyebarkan informasi saluran di seluruh pecahan. Ini menghasilkan peningkatan skalabilitas.
- **Multiplexing I/O yang disempurnakan:** MemoryDB OSS versi 7 memperkenalkan multiplexing I/O yang disempurnakan, yang memberikan peningkatan throughput dan mengurangi latensi untuk beban kerja throughput tinggi yang memiliki banyak koneksi klien bersamaan ke cluster MemoryDB. Misalnya, saat menggunakan sekelompok node r6g.4xlarge dan menjalankan 5200 klien bersamaan, Anda dapat mencapai peningkatan throughput hingga 46% (operasi baca dan tulis per detik) dan latensi P99 menurun hingga 21%, dibandingkan dengan MemoryDB versi 6.

Untuk informasi lebih lanjut tentang rilis Redis OSS 7.0, lihat Catatan Rilis [Redis OSS 7.0 di Redis OSS](#) pada. GitHub

MemoryDB versi 6.2 (ditingkatkan)

MemoryDB memperkenalkan versi berikutnya dari mesin Redis OSS, yang meliputi, dukungan upgrade versi otomatis [Mengautentikasi pengguna dengan Daftar Kontrol Akses \(ACL\)](#), caching sisi klien dan peningkatan operasional yang signifikan.

Redis engine versi 6.2.6 juga memperkenalkan dukungan untuk format asli JavaScript Object Notation (JSON), cara sederhana dan tanpa skema untuk menyandikan kumpulan data kompleks di dalam cluster Redis OSS. Dengan dukungan JSON, Anda dapat memanfaatkan kinerja dan Redis OSS API untuk aplikasi yang beroperasi melalui JSON. Untuk informasi selengkapnya, lihat [Mulai menggunakan JSON](#). Juga termasuk metrik terkait JSON `JsonBasedCmds` yang dimasukkan ke dalam CloudWatch untuk memantau penggunaan tipe data ini. Untuk informasi selengkapnya, lihat [Metrik untuk MemoryDB](#).

Dengan Redis OSS 6, MemoryDB akan menawarkan versi tunggal untuk setiap rilis minor Redis OSS, daripada menawarkan beberapa versi patch. Ini dirancang untuk meminimalkan kebingungan dan ambiguitas karena harus memilih dari beberapa versi minor. MemoryDB juga akan secara otomatis mengelola versi minor dan patch dari cluster yang sedang berjalan, memastikan peningkatan kinerja dan keamanan yang ditingkatkan. Ini akan ditangani melalui saluran pemberitahuan pelanggan standar melalui kampanye pembaruan layanan. Untuk informasi selengkapnya, lihat [Pembaruan layanan di MemoryDB](#).

Jika Anda tidak menentukan versi mesin selama pembuatan, MemoryDB akan secara otomatis memilih versi Redis OSS yang disukai untuk Anda. Di sisi lain, jika Anda menentukan versi mesin dengan menggunakan `6.2`, MemoryDB akan secara otomatis memanggil versi patch pilihan Redis OSS 6.2 yang tersedia.

Misalnya, saat Anda membuat cluster, Anda menyetel `--engine-version` parameternya `6.2`. Cluster akan diluncurkan dengan versi patch pilihan yang tersedia saat ini pada waktu pembuatan. Setiap permintaan dengan nilai versi mesin lengkap akan ditolak, pengecualian akan dilemparkan dan prosesnya akan gagal.

Saat memanggil `DescribeEngineVersions` API, nilai `EngineVersion` parameter akan disetel ke `6.2` dan versi mesin lengkap yang sebenarnya akan dikembalikan di `EnginePatchVersion` bidang.

Untuk informasi lebih lanjut tentang rilis Redis OSS 6.2, lihat [Catatan Rilis Redis 6.2 di Redis OSS](#) pada. GitHub

Meningkatkan versi mesin

MemoryDB secara default secara otomatis mengelola versi patch cluster Anda yang sedang berjalan melalui pembaruan layanan. Anda juga dapat memilih keluar dari upgrade versi auto minor jika Anda menyetel `AutoMinorVersionUpgrade` properti cluster Anda ke `false`. Namun, Anda tidak dapat memilih keluar dari upgrade versi patch otomatis.

Anda dapat mengontrol jika dan kapan perangkat lunak yang sesuai dengan protokol yang mendukung klaster Anda ditingkatkan ke versi baru yang didukung oleh MemoryDB sebelum peningkatan otomatis dimulai. Dengan tingkat kontrol ini, Anda dapat memelihara kompatibilitas dengan versi tertentu, menguji versi baru dengan aplikasi Anda sebelum di-deploy ke sistem produksi, dan melakukan peningkatan versi sesuai syarat dan waktu Anda sendiri.

Anda dapat memulai upgrade versi engine ke cluster Anda dengan cara berikut:

- Dengan memperbaruinya dan menentukan versi mesin baru. Untuk informasi selengkapnya, lihat [Memodifikasi cluster MemoryDB](#).
- Menerapkan pembaruan layanan untuk versi mesin yang sesuai. Untuk informasi selengkapnya, lihat [Pembaruan layanan di MemoryDB](#).

Perhatikan hal berikut:

- Anda dapat meningkatkan ke versi mesin yang lebih baru, tetapi Anda tidak dapat menurunkan ke versi mesin yang lebih lama. Jika Anda ingin menggunakan versi mesin sebelumnya, Anda harus menghapus klaster yang telah ada dan membuatnya lagi dengan versi mesin sebelumnya.
- Kami merekomendasikan upgrade secara berkala ke versi utama terbaru, karena sebagian besar perbaikan besar tidak kembali porting ke versi yang lebih lama. Ketika MemoryDB memperluas ketersediaan ke AWS Wilayah baru, MemoryDB mendukung dua MAJOR.MINOR versi terbaru pada waktu itu untuk Wilayah baru. Misalnya, jika AWS wilayah baru diluncurkan dan versi MAJOR.MINOR MemoryDB terbaru adalah 7.0 dan 6.2, MemoryDB akan mendukung versi 7.0 dan 6.2 di Wilayah baru. AWS Ketika MAJOR.MINOR versi baru dari MemoryDB dirilis, MemoryDB akan terus menambahkan dukungan untuk versi MemoryDB yang baru dirilis. Untuk mempelajari lebih lanjut tentang memilih Wilayah untuk MemoryDB, lihat [Wilayah & titik akhir yang didukung](#).
- Manajemen versi mesin dirancang agar Anda dapat memiliki kontrol sebanyak mungkin terkait cara melakukan patching. Namun, MemoryDB berhak untuk menambal cluster Anda atas nama Anda jika terjadi kerentanan keamanan kritis dalam sistem atau perangkat lunak.

- MemoryDB akan menawarkan satu versi untuk setiap rilis minor Redis OSS, daripada menawarkan beberapa versi patch. Ini dirancang untuk meminimalkan kebingungan dan ambiguitas karena harus memilih dari beberapa versi. MemoryDB juga akan secara otomatis mengelola versi minor dan patch dari cluster yang sedang berjalan, memastikan peningkatan kinerja dan keamanan yang ditingkatkan. Ini akan ditangani melalui saluran pemberitahuan pelanggan standar melalui kampanye pembaruan layanan. Untuk informasi selengkapnya, lihat [Pembaruan layanan di MemoryDB](#).
- Anda dapat meng-upgrade versi cluster Anda dengan downtime minimal. Klaster tersedia untuk operasi baca selama keseluruhan proses peningkatan dan tersedia untuk operasi tulis untuk sebagian besar durasi peningkatan, kecuali selama operasi failover yang berlangsung beberapa detik.
- Kami menyarankan Anda melakukan upgrade mesin selama periode lalu lintas tulis masuk rendah.

Cluster dengan beberapa pecahan diproses dan ditambal sebagai berikut:

- Hanya satu operasi peningkatan yang dilakukan per pecahan kapan saja.
- Di setiap serpihan, semua replika diproses sebelum primer diproses. Jika terdapat lebih sedikit replika dalam serpihan, primer dalam serpihan itu mungkin diproses sebelum replika di serpihan lainnya selesai diproses.
- Di semua serpihan, simpul primer diproses secara berurutan. Hanya satu simpul primer yang ditingkatkan dalam satu waktu.

Topik

- [Cara meningkatkan versi mesin](#)
- [Menyelesaikan peningkatan mesin Redis OSS yang diblokir](#)

Cara meningkatkan versi mesin

Anda memulai upgrade versi ke cluster Anda dengan memodifikasinya menggunakan konsol MemoryDB, API AWS CLI, atau MemoryDB API dan menentukan versi engine yang lebih baru. Untuk informasi selengkapnya, lihat topik berikut.

- [Menggunakan AWS Management Console](#)
- [Menggunakan AWS CLI](#)
- [Menggunakan MemoryDB API](#)

Menyelesaikan peningkatan mesin Redis OSS yang diblokir

Seperti yang ditunjukkan pada tabel berikut, operasi upgrade mesin Redis OSS Anda diblokir jika Anda memiliki operasi penskalaan yang tertunda.

Operasi yang tertunda	Operasi yang diblokir
Penaikan skala	Peningkatan mesin segera
Peningkatan mesin	Penaikan skala segera
Penaikan skala dan peningkatan mesin	Penaikan skala segera
	Peningkatan mesin segera

Mulai menggunakan JSON

MemoryDB mendukung format asli JavaScript Object Notation (JSON), cara sederhana dan tanpa skema untuk menyalin kumpulan data kompleks di dalam cluster Redis OSS. Anda dapat menyimpan dan mengakses data secara native menggunakan format JavaScript Object Notation (JSON) di dalam cluster Redis OSS dan memperbarui data JSON yang disimpan dalam cluster tersebut, tanpa perlu mengelola kode khusus untuk membuat serial dan deserialisasinya.

Selain memanfaatkan Redis OSS API untuk aplikasi yang beroperasi melalui JSON, Anda sekarang dapat secara efisien mengambil dan memperbarui bagian tertentu dari dokumen JSON tanpa perlu memanipulasi seluruh objek, yang dapat meningkatkan kinerja dan mengurangi biaya. Anda juga dapat menelusuri konten dokumen JSON Anda menggunakan kueri JSONPath [bergaya Goessner](#).

Setelah membuat cluster dengan versi mesin yang didukung, tipe data JSON dan perintah terkait secara otomatis tersedia. Ini kompatibel dengan API dan kompatibel dengan RDB dengan versi 2 dari modul RedisJSON, sehingga Anda dapat dengan mudah memigrasikan aplikasi Redis OSS berbasis JSON yang ada ke MemoryDB. Untuk informasi selengkapnya tentang perintah Redis OSS yang didukung, lihat [Perintah yang Didukung](#)

Metrik terkait JSON `JsonBasedCmds` dimasukkan ke dalam CloudWatch untuk memantau penggunaan tipe data ini. Untuk informasi selengkapnya, lihat [Metrik untuk MemoryDB](#).

Note

Untuk menggunakan JSON, Anda harus menjalankan mesin Redis OSS versi 6.2.6 atau yang lebih baru.

Topik

- [Ikhtisar tipe data Redis OSS JSON](#)
- [Perintah yang Didukung](#)

Ikhtisar tipe data Redis OSS JSON

MemoryDB mendukung sejumlah perintah Redis OSS untuk bekerja dengan tipe data JSON. Berikut ini adalah ikhtisar tipe data JSON dan daftar rinci perintah Redis OSS yang didukung.

Terminologi

Istilah	Deskripsi
Dokumen JSON	mengacu pada nilai kunci Redis OSS JSON
Nilai JSON	mengacu pada subset dari Dokumen JSON, termasuk root yang mewakili seluruh dokumen. Nilai bisa berupa wadah atau entri dalam wadah
Elemen JSON	setara dengan nilai JSON

Standar JSON yang didukung

Format JSON sesuai dengan standar pertukaran data JSON [RFC 7159](#) dan [ECMA-404](#). Mendukung UTF-8 [Unicode](#) pada teks JSON.

Elemen root

Elemen root dapat berupa jenis data JSON apa pun. Perhatikan bahwa di RFC 4627 sebelumnya, hanya objek atau array yang diizinkan sebagai nilai root. Sejak pembaruan ke RFC 7159, root dokumen JSON dapat berupa jenis data JSON apa pun.

Batas ukuran dokumen

Dokumen JSON disimpan secara internal dalam format yang dioptimalkan untuk akses cepat dan modifikasi. Format ini biasanya menghasilkan lebih banyak memori daripada representasi serial yang setara dari dokumen yang sama. Konsumsi memori oleh satu dokumen JSON dibatasi hingga 64MB, yang merupakan ukuran struktur data dalam memori, bukan string JSON. Jumlah memori yang dikonsumsi oleh dokumen JSON dapat diperiksa dengan menggunakan perintah. `JSON.DEBUG MEMORY`

ACL JSON

- Jenis data JSON sepenuhnya terintegrasi ke dalam kemampuan [Redis Access Control Lists](#) (ACL). Mirip dengan kategori per-tipe data yang ada (`@string`, `@hash`, dll.) Kategori baru `@json` ditambahkan untuk menyederhanakan pengelolaan akses ke perintah dan data JSON. Tidak ada perintah Redis OSS lain yang ada sebagai anggota kategori `@json`. Semua perintah JSON memberlakukan batasan dan izin ruang kunci atau perintah apa pun.
- Ada lima kategori Redis OSS ACL yang diperbarui untuk menyertakan perintah JSON baru: `@read`, `@write`, `@fast`, `@slow` dan `@admin`. Tabel di bawah ini menunjukkan pemetaan perintah JSON ke kategori yang sesuai.

ACL

Perintah JSON	@read	@write	@fast	@slow	@admin
JSON.ARRAPPEND		y	y		
JSON.ARRINDEX	y		y		
JSON.ARRINSERT		y	y		

Perintah JSON	@read	@write	@fast	@slow	@admin
JSON.ARRLEN	y		y		
JSON.ARRPOP		y	y		
JSON.ARRTRIM		y	y		
JSON.CLEAR		y	y		
JSON.DEBUG	y			y	y
JSON.DEL		y	y		
JSON.FORGET		y	y		
JSON.GET	y		y		
JSON.MGET	y		y		
JSON.NUMINCRBY		y	y		
JSON.NUMMULTBY		y	y		
JSON.OBJECTS	y		y		
JSON.OBJECTLEN	y		y		
JSON.RESP	y		y		

Perintah JSON	@read	@write	@fast	@slow	@admin
JSON.SET		y		y	
JSON.STRAPPEND		y	y		
JSON.STRLEN	y		y		
JSON.STRLEN	y		y		
JSON.TOGGLE		y	y		
JSON.TYPE	y		y		
JSON.NUMINCRBY		y	y		

Batas kedalaman sarang

Ketika objek atau array JSON memiliki elemen yang merupakan objek atau array JSON lain, objek dalam atau array dianggap "bersarang" dalam objek luar atau array. Batas kedalaman sarang maksimum adalah 128. Setiap percobaan untuk membuat dokumen yang berisi kedalaman bersarang lebih dari 128 akan ditolak dengan kesalahan.

Sintaksis perintah

Kebanyakan perintah memerlukan nama kunci Redis OSS sebagai argumen pertama. Beberapa perintah juga memiliki argumen jalur. Argumen path default ke root jika opsional dan tidak disediakan.

Notasi:

- Argumen yang diperlukan diapit dalam tanda kurung sudut, mis. <key>
- Argumen opsional terlampir dalam tanda kurung siku, misalnya [path]
- Argumen opsional tambahan ditunjukkan oleh..., misalnya [json...]

Sintaksis jalur

JSON-Redis OSS mendukung dua jenis sintaks jalur:

- Sintaks yang disempurnakan - Mengikuti sintaks JSONPath yang dijelaskan oleh [Goessner](#), seperti yang ditunjukkan pada tabel di bawah ini. Kami telah menyusun ulang dan mengubah deskripsi dalam tabel agar jelas.
- Sintaksis terbatas – Memiliki kemampuan kueri terbatas.

Note

Hasil dari beberapa perintah sensitif jenis sintaks jalur yang digunakan.

Jika jalur kueri diawali dengan '\$', kueri tersebut menunjukkan penggunaan sintaksis yang ditingkatkan. Jika tidak, maka kueri tersebut menggunakan sintaksis terbatas.

Sintaks yang Ditingkatkan

Simbol/Ekspresi	Deskripsi
\$	elemen root
. atau []	operator anak
..	keturunan rekursif
*	wildcard. Semua elemen dalam sebuah objek atau array.
[]	operator subskrip array. Indeks berbasis 0.
[.]	operator serikat
[start:end:step]	operator irisan array
?()	menerapkan ekspresi filter (script) ke array atau objek saat ini

Simbol/Ekspresi	Deskripsi
()	ekspresi filter
@	digunakan dalam ekspresi filter mengacu pada node saat ini sedang diproses
==	sama dengan, digunakan dalam ekspresi filter.
!=	tidak sama dengan, digunakan dalam ekspresi filter.
>	lebih besar dari, digunakan dalam ekspresi filter.
>=	lebih besar dari atau sama dengan, digunakan dalam ekspresi filter.
<	kurang dari, digunakan dalam ekspresi filter.
<=	kurang dari atau sama dengan, digunakan dalam ekspresi filter.
&&	logis AND, digunakan untuk menggabungkan beberapa ekspresi filter.
	logis OR, digunakan untuk menggabungkan beberapa ekspresi filter.

Contoh

Contoh di bawah ini dibangun di atas contoh data XHTML [Goessner](#), yang telah kami modifikasi dengan menambahkan bidang tambahan.

```
{ "store": {
  "book": [
    { "category": "reference",
      "author": "Nigel Rees",
      "title": "Sayings of the Century",
      "price": 8.95,
```

```

    "in-stock": true,
    "sold": true
  },
  { "category": "fiction",
    "author": "Evelyn Waugh",
    "title": "Sword of Honour",
    "price": 12.99,
    "in-stock": false,
    "sold": true
  },
  { "category": "fiction",
    "author": "Herman Melville",
    "title": "Moby Dick",
    "isbn": "0-553-21311-3",
    "price": 8.99,
    "in-stock": true,
    "sold": false
  },
  { "category": "fiction",
    "author": "J. R. R. Tolkien",
    "title": "The Lord of the Rings",
    "isbn": "0-395-19395-8",
    "price": 22.99,
    "in-stock": false,
    "sold": false
  }
],
"bicycle": {
  "color": "red",
  "price": 19.95,
  "in-stock": true,
  "sold": false
}
}
}

```

Jalur	Deskripsi
<code>\$.store.book[*].author</code>	penulis semua buku di toko
<code>\$.author</code>	semua penulis

Jalur	Deskripsi
<code>\$.store.*</code>	semua anggota toko
<code>\$["store"].*</code>	semua anggota toko
<code>\$.store..price</code>	harga semua yang ada di toko
<code>\$..*</code>	semua anggota rekursif dari struktur JSON
<code>\$.book[*]</code>	semua buku
<code>\$.book[0]</code>	buku pertama
<code>\$.book[-1]</code>	buku terakhir
<code>\$.book[0:2]</code>	Dua buku pertama
<code>\$.book[0,1]</code>	Dua buku pertama
<code>\$.book[0:4]</code>	buku dari indeks 0 hingga 3 (indeks akhir tidak inklusif)
<code>\$.book[0:4:2]</code>	buku pada indeks 0, 2
<code>\$.book[?(@.isbn)]</code>	semua buku dengan nomor isbn
<code>\$.book[?(@.price<10)]</code>	Semua buku lebih murah dari \$10
<code>'\$.book[?(@.price < 10)]'</code>	Semua buku lebih murah dari \$10. (Jalur harus dikutip jika berisi spasi putih)
<code>'\$.book[?(@.["price"] < 10)]'</code>	Semua buku lebih murah dari \$10
<code>'\$.book[?(@.["price"] < 10)]'</code>	Semua buku lebih murah dari \$10
<code>\$.book[?(@.price>=10&&@.price<=100)]</code>	semua buku dalam kisaran harga \$10 hingga \$100, inklusif

Jalur	Deskripsi
'\$.book[?(@.price>=10 && @.price<=100)]'	semua buku dalam kisaran harga \$10 hingga \$100, inklusif. (Jalur harus dikutip jika berisi spasi putih)
\$.book[?(@.sold==true @.in-stock==false)]	semua buku terjual atau kehabisan stok
'\$.book[?(@.sold == true @.in-stock == false)]'	Semua buku terjual atau kehabisan stok. (Jalur harus dikutip jika berisi spasi putih)
'\$.store.book[?(@.["category"] == "fiction")]	semua buku dalam kategori fiksi
'\$.store.book[?(@.["category"] != "fiction")]	semua buku dalam kategori non-fiksi

Contoh ekspresi filter lainnya:

```
127.0.0.1:6379> JSON.SET k1 . '{"books": [{"price":5,"sold":true,"in-stock":true,"title":"foo"}, {"price":15,"sold":false,"title":"abc"}]}'
OK
127.0.0.1:6379> JSON.GET k1 $.books[?(@.price>1&&@.price<20&&@.in-stock)]
"[{"price":5,"sold":true,"in-stock":true,"title":"foo"}]"
127.0.0.1:6379> JSON.GET k1 '$.books[?(@.price>1 && @.price<20 && @.in-stock)]'
"[{"price":5,"sold":true,"in-stock":true,"title":"foo"}]"
127.0.0.1:6379> JSON.GET k1 '$.books[?((@.price>1 && @.price<20) && (@.sold==false))]'
"[{"price":15,"sold":false,"title":"abc"}]"
127.0.0.1:6379> JSON.GET k1 '$.books[?(@.title == "abc")]'
[{"price":15,"sold":false,"title":"abc"}]

127.0.0.1:6379> JSON.SET k2 . '[1,2,3,4,5]'
127.0.0.1:6379> JSON.GET k2 $.*.[?(@>2)]
"[3,4,5]"
127.0.0.1:6379> JSON.GET k2 '$.*.[?(@ > 2)]'
"[3,4,5]"

127.0.0.1:6379> JSON.SET k3 . '[true,false,true,false,null,1,2,3,4]'
OK
127.0.0.1:6379> JSON.GET k3 $.*.[?(@==true)]
"[true,true]"
127.0.0.1:6379> JSON.GET k3 '$.*.[?(@ == true)]'
"[true,true]"
```

```
127.0.0.1:6379> JSON.GET k3 $.*.[?(@>1)]
"[2,3,4]"
127.0.0.1:6379> JSON.GET k3 '$.*.[?(@ > 1)]'
"[2,3,4]"
```

Sintaksis terbatas

Simbol/Ekspresi	Deskripsi
. atau []	operator anak
[]	operator subskrip array. Indeks berbasis 0.

Contoh

Jalur	Deskripsi
.store.book[0].author	penulis buku pertama
.store.book[-1].author	penulis buku terakhir
.address.city	nama kota
["store"]["book"][0]["title"]	Judul buku pertama
["store"]["book"][-1]["title"]	Judul buku terakhir

Note

Semua konten [Goessner](#) yang dikutip dalam dokumentasi ini diatur dengan [Lisensi Creative Commons](#).

Awalan kesalahan umum

Setiap pesan kesalahan memiliki awalan. Berikut ini adalah daftar awalan kesalahan umum:

Awalan	Deskripsi
ERR	kesalahan umum
LIMIT	batas ukuran melebihi kesalahan. misalnya, batas ukuran dokumen atau batas kedalaman bersarang terlampaui
NONEXISTENT	kunci atau jalur tidak ada
OUTOFBOUNDARIES	indeks array di luar batas
SYNTAXERR	kesalahan sintaks
WRONGTYPE	jenis nilai yang salah

Metrik terkait JSON

Tersedia metrik info JSON berikut:

Info	Deskripsi
json_total_memory_bytes	total memori yang dialokasikan ke objek JSON
json_num_documents	jumlah total dokumen di Redis OSS

Untuk menanyakan metrik inti, jalankan perintah Redis OSS:

```
info json_core_metrics
```

Bagaimana MemoryDB berinteraksi dengan JSON

Berikut ini menggambarkan bagaimana MemoryDB berinteraksi dengan tipe data JSON.

Prasyarat operator

Saat mengevaluasi ekspresi bersyarat untuk pemfilteran, `&&` diutamakan terlebih dahulu, lalu `||` dievaluasi, seperti yang umum di sebagian besar bahasa. Operasi di dalam tanda kurung akan dieksekusi terlebih dahulu.

Perilaku batas bersarang jalur maksimum

Batas bersarang jalur maksimum MemoryDB adalah 128. Jadi nilai seperti `$.a.b.c.d...` hanya bisa mencapai 128 tingkat.

Menangani nilai numerik

JSON tidak memiliki tipe data terpisah untuk bilangan bulat dan floating point. Semuanya disebut angka.

Ketika nomor JSON diterima, itu disimpan dalam salah satu dari dua format. Jika nomor cocok dengan integer bertanda 64-bit, maka itu dikonversi ke format itu; jika tidak, itu disimpan sebagai string. Operasi aritmatika pada dua nomor JSON (misalnya `JSON.NUMINCRBY` dan `JSON.NUMMULTBY`) berusaha mempertahankan presisi sebanyak mungkin. Jika dua operan dan nilai yang dihasilkan cocok dengan integer bertanda 64-bit, maka aritmatika integer dilakukan. Jika tidak, operan input diubah menjadi nomor floating point presisi ganda IEEE 64-bit, operasi aritmatika dilakukan, dan hasilnya diubah kembali menjadi string.

Perintah aritmetika `NUMINCRBY` dan `NUMMULTBY`:

- Jika kedua angka adalah bilangan bulat, dan hasilnya berada di luar kisaran `int64`, secara otomatis akan menjadi nomor floating point presisi ganda.
- Jika setidaknya salah satu angka adalah floating point, hasilnya akan menjadi nomor floating point presisi ganda.
- Jika hasilnya melebihi kisaran ganda, perintah akan mengembalikan `OVERFLOW` kesalahan.

Note

Sebelum mesin Redis OSS versi 6.2.6.R2 ketika nomor JSON diterima pada input, itu diubah menjadi salah satu dari dua representasi biner internal: integer bertanda 64-bit atau titik mengambang presisi ganda IEEE 64-bit. String asli dan semua formatnya tidak dipertahankan. Jadi, ketika angka dihasilkan sebagai bagian dari respons JSON, angka tersebut dikonversi dari representasi biner internal ke string yang dapat dicetak yang menggunakan aturan pemformatan generik. Aturan-aturan ini dapat menghasilkan string yang berbeda dari string yang diterima.

- Jika kedua angka ini adalah integer dan hasilnya berada di luar rentang `int64`, maka angka tersebut secara otomatis akan menjadi angka floating point presisi ganda IEEE 64-bit.

- Jika minimal salah satu angkanya adalah floating point, hasilnya adalah angka floating point presisi ganda IEEE 64-bit.
- Jika hasilnya melebihi rentang IEEE 64-bit ganda, perintah ini menampilkan kesalahan OVERFLOW.

Untuk daftar lengkap perintah yang tersedia, lihat [Perintah yang Didukung](#).

Evaluasi sintaksis yang ketat

MemoryDB tidak mengizinkan jalur JSON dengan sintaksis yang tidak valid, bahkan jika subset dari jalur berisi jalur yang valid. Hal ini dimaksudkan untuk menjaga perilaku yang benar bagi pelanggan kami.

Perintah yang Didukung

Perintah Redis OSS JSON berikut didukung:

Topik

- [JSON.ARRAPPEND](#)
- [JSON.ARRINDEX](#)
- [JSON.ARRINSERT](#)
- [JSON.ARRLEN](#)
- [JSON.ARRPOP](#)
- [JSON.ARRTRIM](#)
- [JSON.CLEAR](#)
- [JSON.DEBUG](#)
- [JSON.DEL](#)
- [JSON.FORGET](#)
- [JSON.GET](#)
- [JSON.MGET](#)
- [JSON.NUMINCRBY](#)
- [JSON.NUMMULTBY](#)
- [JSON.OBJLEN](#)
- [JSON.OBJKEYS](#)

- [JSON.RESP](#)
- [JSON.SET](#)
- [JSON.STRAPPEND](#)
- [JSON.STRLLEN](#)
- [JSON.TOGGLE](#)
- [JSON.TYPE](#)

JSON.ARRAPPEND

Tambahkan satu atau lebih nilai ke nilai array di jalur.

Sintaks

```
JSON.ARRAPPEND <key> <path> <json> [json ...]
```

- kunci (wajib) - Kunci Redis OSS dari jenis dokumen JSON
- jalur (diperlukan) - jalur JSON
- json (wajib) - nilai JSON yang akan ditambahkan ke array

Nilai yang ditampilkan

Jika jalur adalah sintaksis yang ditingkatkan:

- Array bilangan bulat, mewakili panjang baru dari array di setiap jalur.
- Jika nilai bukan array, nilai yang akan dikembalikan adalah kosong.
- Kesalahan SYNTAXERR jika salah satu argumen input json bukan string JSON yang valid.
- Kesalahan NONEXISTENT jika jalur tidak ada.

Jika jalur adalah sintaksis terbatas:

- Integer, panjang baru array.
- Jika memilih beberapa nilai array, perintah ini mengembalikan panjang baru dari array yang terakhir diperbarui.
- Kesalahan WRONGTYPE jika nilai di jalur bukan array.
- Kesalahan SYNTAXERR jika salah satu argumen input json bukan string JSON yang valid.

- Kesalahan NONEXISTENT jika jalur tidak ada.

Contoh

Sintaksis jalur yang ditingkatkan:

```
127.0.0.1:6379> JSON.SET k1 . '[[[], ["a"], ["a", "b"]]'
OK
127.0.0.1:6379> JSON.ARRAPPEND k1 $[*] '"c"'
1) (integer) 1
2) (integer) 2
3) (integer) 3
127.0.0.1:6379> JSON.GET k1
"[[\"c\"],[\"a\",\"c\"],[\"a\",\"b\",\"c\"]]"
```

Sintaksis jalur terbatas:

```
127.0.0.1:6379> JSON.SET k1 . '[[[], ["a"], ["a", "b"]]'
OK
127.0.0.1:6379> JSON.ARRAPPEND k1 [-1] '"c"'
(integer) 3
127.0.0.1:6379> JSON.GET k1
"[[],[\"a\"],[\"a\",\"b\",\"c\"]]"
```

JSON.ARRINDEX

Cari kejadian pertama dari nilai JSON skalar dalam array di jalur.

- Kesalahan di luar jangkauan diatasi dengan membulatkan indeks ke awal dan akhir array.
- Jika awal > akhir, mengembalikan -1 (tidak ditemukan).

Sintaks

```
JSON.ARRINDEX <key> <path> <json-scalar> [start [end]]
```

- kunci (wajib) - Kunci Redis OSS dari jenis dokumen JSON
- jalur (diperlukan) - jalur JSON

- **json-scalar (wajib)** — nilai skalar untuk dicari; Skalar JSON mengacu pada nilai yang bukan objek atau array. yaitu, String, number, boolean dan null adalah nilai skalar.
- **mulai (opsional)** - indeks mulai, inklusif. Default ke 0 jika tidak disediakan.
- **akhir (opsional)** - indeks akhir, eksklusif. Default ke 0 jika tidak disediakan, yang berarti elemen terakhir disertakan. 0 atau -1 berarti elemen terakhir disertakan.

Nilai yang ditampilkan

Jika jalur adalah sintaksis yang ditingkatkan:

- Array integer. Setiap nilai adalah indeks dari elemen yang cocok dalam array di jalur. Nilainya -1 jika tidak ditemukan.
- Jika nilai bukan array, nilai yang akan dikembalikan adalah kosong.

Jika jalur adalah sintaksis terbatas:

- Integer, indeks elemen yang cocok, atau -1 jika tidak ditemukan.
- Kesalahan `WRONGTYPE` jika nilai di jalur bukan array.

Contoh

Sintaksis jalur yang ditingkatkan:

```
127.0.0.1:6379> JSON.SET k1 . '[[[], ["a"], ["a", "b"], ["a", "b", "c"]]'
OK
127.0.0.1:6379> JSON.ARRINDEX k1 $[*] '"b"'
1) (integer) -1
2) (integer) -1
3) (integer) 1
4) (integer) 1
```

Sintaksis jalur terbatas:

```
127.0.0.1:6379> JSON.SET k1 . '{"children": ["John", "Jack", "Tom", "Bob", "Mike"]}'
OK
127.0.0.1:6379> JSON.ARRINDEX k1 .children '"Tom"'
(integer) 2
```

JSON.ARRINSERT

Masukkan satu atau lebih nilai ke dalam nilai array di jalur sebelum indeks.

Sintaks

```
JSON.ARRINSERT <key> <path> <index> <json> [json ...]
```

- kunci (wajib) - Kunci Redis OSS dari jenis dokumen JSON
- jalur (diperlukan) - jalur JSON
- index (required) — indeks array sebelum nilai yang dimasukkan.
- json (wajib) - nilai JSON yang akan ditambahkan ke array

Nilai yang ditampilkan

Jika jalur adalah sintaksis yang ditingkatkan:

- Array bilangan bulat, mewakili panjang baru dari array di setiap jalur.
- Jika nilai adalah array kosong, nilai yang akan dikembalikan adalah kosong.
- Jika nilai bukan array, nilai yang akan dikembalikan adalah kosong.
- Kesalahan `OUTOFBOUNDARIES` jika argumen indeks di luar batas.

Jika jalur adalah sintaksis terbatas:

- Integer, panjang baru array.
- Kesalahan `WRONGTYPE` jika nilai di jalur bukan array.
- Kesalahan `OUTOFBOUNDARIES` jika argumen indeks di luar batas.

Contoh

Sintaksis jalur yang ditingkatkan:

```
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"]]'  
OK  
127.0.0.1:6379> JSON.ARRINSERT k1 $[*] 0 '"c"'
```

```

1) (integer) 1
2) (integer) 2
3) (integer) 3
127.0.0.1:6379> JSON.GET k1
"[[\"c\"],[\"c\",\"a\"],[\"c\",\"a\",\"b\"]]"

```

Sintaksis jalur terbatas:

```

127.0.0.1:6379> JSON.SET k1 . '[[[], ["a"], ["a", "b"]]'
OK
127.0.0.1:6379> JSON.ARRINSERT k1 . 0 "c"
(integer) 4
127.0.0.1:6379> JSON.GET k1
"[[\"c\"],[], [\"a\"],[\"a\",\"b\"]]"

```

JSON.ARRLEN

Dapatkan panjang nilai array di jalur.

Sintaks

```
JSON.ARRLEN <key> [path]
```

- kunci (wajib) - Kunci Redis OSS dari jenis dokumen JSON
- path (opsional) - jalur JSON. Default ke root jika tidak disediakan

Nilai yang ditampilkan

Jika jalur adalah sintaksis yang ditingkatkan:

- Array bilangan bulat, mewakili panjang array di setiap jalur.
- Jika nilai bukan array, nilai yang akan dikembalikan adalah kosong.
- Kosong jika kunci dokumen tidak ada.

Jika jalur adalah sintaksis terbatas:

- Array string massal. Setiap elemen adalah nama kunci dalam objek.

- Integer, panjang array.
- Jika memilih beberapa objek, perintah ini mengembalikan panjang array pertama.
- Kesalahan `WRONGTYPE` jika nilai di jalur bukan array.
- Kesalahan `WRONGTYPE` jika jalur tidak ada.
- Kosong jika kunci dokumen tidak ada.

Contoh

Sintaksis jalur yang ditingkatkan:

```
127.0.0.1:6379> JSON.SET k1 . '[[[], [\"a\"], [\"a\", \"b\"], [\"a\", \"b\", \"c\"]]'
(error) SYNTAXERR Failed to parse JSON string due to syntax error
127.0.0.1:6379> JSON.SET k1 . '[[[], [\"a\"], [\"a\", \"b\"], [\"a\", \"b\", \"c\"]]'
OK
127.0.0.1:6379> JSON.ARRLEN k1 $[*]
1) (integer) 0
2) (integer) 1
3) (integer) 2
4) (integer) 3

127.0.0.1:6379> JSON.SET k2 . '[[[], \"a\", [\"a\", \"b\"], [\"a\", \"b\", \"c\"], 4]'
OK
127.0.0.1:6379> JSON.ARRLEN k2 $[*]
1) (integer) 0
2) (nil)
3) (integer) 2
4) (integer) 3
5) (nil)
```

Sintaksis jalur terbatas:

```
127.0.0.1:6379> JSON.SET k1 . '[[[], [\"a\"], [\"a\", \"b\"], [\"a\", \"b\", \"c\"]]'
OK
127.0.0.1:6379> JSON.ARRLEN k1 [*]
(integer) 0
127.0.0.1:6379> JSON.ARRLEN k1 $[3]
1) (integer) 3

127.0.0.1:6379> JSON.SET k2 . '[[[], \"a\", [\"a\", \"b\"], [\"a\", \"b\", \"c\"], 4]'
```

```
OK
127.0.0.1:6379> JSON.ARRLEN k2 [*]
(integer) 0
127.0.0.1:6379> JSON.ARRLEN k2 $[1]
1) (nil)
127.0.0.1:6379> JSON.ARRLEN k2 $[2]
1) (integer) 2
```

JSON.ARRPOP

Hapus dan kembalikan elemen pada indeks dari array. Popping array kosong akan menampilkan null.

Sintaks

```
JSON.ARRPOP <key> [path [index]]
```

- kunci (wajib) - Kunci Redis OSS dari jenis dokumen JSON
- path (opsional) - jalur JSON. Default ke root jika tidak disediakan
- index (opsional) — posisi dalam array untuk mulai muncul dari.
 - Default ke -1 jika tidak disediakan, yang berarti elemen terakhir.
 - Nilai negatif berarti posisi dari elemen terakhir.
 - Indeks di luar batas akan dibulatkan ke batas array masing-masing.

Nilai yang ditampilkan

Jika jalur adalah sintaksis yang ditingkatkan:

- Array string massal, mewakili nilai yang muncul di setiap jalur.
- Jika nilai adalah array kosong, nilai yang akan dikembalikan adalah kosong.
- Jika nilai bukan array, nilai yang akan dikembalikan adalah kosong.

Jika jalur adalah sintaksis terbatas:

- String massal, mewakili nilai JSON yang muncul
- Null jika array kosong.
- Kesalahan WRONGTYPE jika nilai di jalur bukan array.

Contoh

Sintaksis jalur yang ditingkatkan:

```
127.0.0.1:6379> JSON.SET k1 . '[[[], ["a"], ["a", "b"]]]'
OK
127.0.0.1:6379> JSON.ARRPOP k1 $[*]
1) (nil)
2) "\"a\""
3) "\"b\""
127.0.0.1:6379> JSON.GET k1
"[[[], [], [\"a\"]]]"
```

Sintaksis jalur terbatas:

```
127.0.0.1:6379> JSON.SET k1 . '[[[], ["a"], ["a", "b"]]]'
OK
127.0.0.1:6379> JSON.ARRPOP k1
"[\\"a\\", \\"b\"]"
127.0.0.1:6379> JSON.GET k1
"[[[], [\"a\"]]"

127.0.0.1:6379> JSON.SET k2 . '[[[], ["a"], ["a", "b"]]]'
OK
127.0.0.1:6379> JSON.ARRPOP k2 . 0
"[]"
127.0.0.1:6379> JSON.GET k2
"[[\\"a\\", [\"a\\", \\"b\"]]"
```

JSON.ARRTRIM

Potong array di jalur sehingga menjadi subarray [start, end], keduanya inklusif.

- Jika array kosong, tidak melakukan apa pun, mengembalikan 0.
- Jika awal < 0, perlakukan itu sebagai 0.
- Jika akhir >= ukuran (ukuran array), perlakukan itu sebagai ukuran-1.
- Jika awal >= ukuran atau awal > akhir, mengosongkan array dan menampilkan 0.

Sintaks

```
JSON.ARRINSERT <key> <path> <start> <end>
```

- kunci (wajib) - Kunci Redis OSS dari jenis dokumen JSON
- jalur (diperlukan) - jalur JSON
- mulai (wajib) - indeks mulai, inklusif.
- akhir (wajib) - indeks akhir, inklusif.

Nilai yang ditampilkan

Jika jalur adalah sintaksis yang ditingkatkan:

- Array bilangan bulat, mewakili panjang baru dari array di setiap jalur.
- Jika nilai adalah array kosong, nilai yang akan dikembalikan adalah kosong.
- Jika nilai bukan array, nilai yang akan dikembalikan adalah kosong.
- Kesalahan `OUTOFBOUNDARIES` jika argumen indeks di luar batas.

Jika jalur adalah sintaksis terbatas:

- Integer, panjang baru array.
- Null jika array kosong.
- Kesalahan `WRONGTYPE` jika nilai di jalur bukan array.
- Kesalahan `OUTOFBOUNDARIES` jika argumen indeks di luar batas.

Contoh

Sintaksis jalur yang ditingkatkan:

```
127.0.0.1:6379> JSON.SET k1 . '[[[], ["a"], ["a", "b"], ["a", "b", "c"]]]'
OK
127.0.0.1:6379> JSON.ARRTRIM k1 $[*] 0 1
1) (integer) 0
2) (integer) 1
3) (integer) 2
4) (integer) 2
127.0.0.1:6379> JSON.GET k1
"[[],[\"a\"],[\"a\",\"b\"],[\"a\",\"b\"]]"
```

Sintaksis jalur terbatas:

```
127.0.0.1:6379> JSON.SET k1 . '{"children": ["John", "Jack", "Tom", "Bob", "Mike"]}'
OK
127.0.0.1:6379> JSON.ARRTRIM k1 .children 0 1
(integer) 2
127.0.0.1:6379> JSON.GET k1 .children
"[\\"John\\",\\"Jack\\""]"
```

JSON.CLEAR

Bersihkan array atau objek di jalan.

Sintaks

```
JSON.CLEAR <key> [path]
```

- kunci (wajib) - Kunci Redis OSS dari jenis dokumen JSON
- path (opsional) - jalur JSON. Default ke root jika tidak disediakan

Nilai yang ditampilkan

- Integer, jumlah kontainer dihapus.
- Menghapus array kosong atau akun objek untuk 0 kontainer dihapus.

Note

Prior ke Redis OSS versi 6.2.6.R2, membersihkan array kosong atau akun objek untuk 1 kontainer dihapus.

- Menghapus nilai non-kontainer mengembalikan 0.
- Jika tidak ada array atau nilai objek terletak di jalan, perintah mengembalikan 0.

Contoh

```
127.0.0.1:6379> JSON.SET k1 . '[[[], [0], [0,1], [0,1,2], 1, true, null, "d"]'
```

```

OK
127.0.0.1:6379> JSON.CLEAR k1 $[*]
(integer) 6
127.0.0.1:6379> JSON.CLEAR k1 $[*]
(integer) 0
127.0.0.1:6379> JSON.SET k2 . '{"children": ["John", "Jack", "Tom", "Bob", "Mike"]}'
OK
127.0.0.1:6379> JSON.CLEAR k2 .children
(integer) 1
127.0.0.1:6379> JSON.GET k2 .children
"[]"

```

JSON.DEBUG

Laporkan informasi. Subperintah yang didukung adalah:

- **MEMORY <key>[path]** — melaporkan penggunaan memori dalam byte dari nilai JSON. Jalur akan diatur secara default ke root jika tidak disediakan.
- **<key>DEPTH [path]** — Melaporkan kedalaman jalur maksimum dokumen JSON.

Note

Subperintah ini hanya tersedia menggunakan mesin Redis OSS versi 6.2.6.R2 atau yang lebih baru.

- **FIELDS <key>[path]** — laporkan jumlah bidang di jalur dokumen yang ditentukan. Jalur akan diatur secara default ke root jika tidak disediakan. Setiap nilai JSON non-kontainer dihitung sebagai satu bidang. Objek dan array secara rekursif menghitung satu bidang untuk masing-masing nilai JSON. Setiap nilai kontainer, kecuali kontainer root, dihitung sebagai satu bidang tambahan.
- **HELP** — cetak pesan bantuan dari perintah.

Sintaks

```
JSON.DEBUG <subcommand & arguments>
```

Tergantung pada subperintah:

MEMORY

- Jika jalur adalah sintaksis yang ditingkatkan:
 - mengembalikan array bilangan bulat, mewakili ukuran memori (dalam byte) dari nilai JSON di setiap jalur.
 - mengembalikan array kosong jika kunci Redis OSS tidak ada.
- Jika jalur adalah sintaksis terbatas:
 - mengembalikan integer, ukuran memori nilai JSON dalam byte.
 - mengembalikan null jika kunci Redis OSS tidak ada.

KEDALAMAN

- Mengembalikan integer yang mewakili kedalaman jalur maksimum dokumen JSON.
- Mengembalikan null jika kunci Redis OSS tidak ada.

FIELD

- Jika jalur adalah sintaksis yang ditingkatkan:
 - mengembalikan array bilangan bulat, mewakili jumlah bidang nilai JSON di setiap jalur.
 - mengembalikan array kosong jika kunci Redis OSS tidak ada.
- Jika jalur adalah sintaksis terbatas:
 - mengembalikan integer, jumlah bidang nilai JSON.
 - mengembalikan null jika kunci Redis OSS tidak ada.

HELP — mengembalikan array pesan bantuan.

Contoh

Sintaksis jalur yang ditingkatkan:

```
127.0.0.1:6379> JSON.SET k1 . '[1, 2.3, "foo", true, null, {}, [], {"a":1, "b":2}, [1,2,3]]'
OK
127.0.0.1:6379> JSON.DEBUG MEMORY k1 $[*]
1) (integer) 16
2) (integer) 16
3) (integer) 19
4) (integer) 16
```

```

5) (integer) 16
6) (integer) 16
7) (integer) 16
8) (integer) 50
9) (integer) 64
127.0.0.1:6379> JSON.DEBUG FIELDS k1 $[*]
1) (integer) 1
2) (integer) 1
3) (integer) 1
4) (integer) 1
5) (integer) 1
6) (integer) 0
7) (integer) 0
8) (integer) 2
9) (integer) 3

```

Sintaksis jalur terbatas:

```

127.0.0.1:6379> JSON.SET k1 .
{"firstName":"John","lastName":"Smith","age":27,"weight":135.25,"isAlive":true,"address":
{"street":"21 2nd Street","city":"New
York","state":"NY","zipcode":"10021-3100"},"phoneNumbers":
[{"type":"home","number":"212 555-1234"}, {"type":"office","number":"646
555-4567"}], "children":[], "spouse":null}'
OK
127.0.0.1:6379> JSON.DEBUG MEMORY k1
(integer) 632
127.0.0.1:6379> JSON.DEBUG MEMORY k1 .phoneNumbers
(integer) 166

127.0.0.1:6379> JSON.DEBUG FIELDS k1
(integer) 19
127.0.0.1:6379> JSON.DEBUG FIELDS k1 .address
(integer) 4

127.0.0.1:6379> JSON.DEBUG HELP
1) JSON.DEBUG MEMORY <key> [path] - report memory size (bytes) of the JSON element.
   Path defaults to root if not provided.
2) JSON.DEBUG FIELDS <key> [path] - report number of fields in the JSON element. Path
   defaults to root if not provided.
3) JSON.DEBUG HELP - print help message.

```

JSON.DEL

Hapus nilai JSON di jalur dalam kunci dokumen. Jika jalurnya adalah root, itu setara dengan menghapus kunci dari Redis OSS.

Sintaks

```
JSON.DEL <key> [path]
```

- kunci (wajib) - Kunci Redis OSS dari jenis dokumen JSON
- path (opsional) - jalur JSON. Default ke root jika tidak disediakan

Nilai yang ditampilkan

- Jumlah elemen yang dihapus.
- 0 jika kunci Redis OSS tidak ada.
- 0 jika jalur JSON tidak valid atau tidak ada.

Contoh

Sintaksis jalur yang ditingkatkan:

```
127.0.0.1:6379> JSON.SET k1 . '{"a":{}, "b":{"a":1}, "c":{"a":1, "b":2}, "d":{"a":1, "b":2, "c":3}, "e": [1,2,3,4,5]}'
OK
127.0.0.1:6379> JSON.DEL k1 $.d.*
(integer) 3
127.0.0.1:6379> JSON.GET k1
"{\"a\":{},\"b\":{\"a\":1},\"c\":{\"a\":1,\"b\":2},\"d\":{},\"e\":[1,2,3,4,5]}"
127.0.0.1:6379> JSON.DEL k1 $.e[*]
(integer) 5
127.0.0.1:6379> JSON.GET k1
"{\"a\":{},\"b\":{\"a\":1},\"c\":{\"a\":1,\"b\":2},\"d\":{},\"e\":[]}"
```

Sintaksis jalur terbatas:

```

127.0.0.1:6379> JSON.SET k1 . '{"a":{}, "b":{"a":1}, "c":{"a":1, "b":2}, "d":{"a":1,
"b":2, "c":3}, "e": [1,2,3,4,5]}'
OK
127.0.0.1:6379> JSON.DEL k1 .d.*
(integer) 3
127.0.0.1:6379> JSON.GET k1
"{\"a\":{},\"b\":{\"a\":1},\"c\":{\"a\":1,\"b\":2},\"d\":{},\"e\":[1,2,3,4,5]}"
127.0.0.1:6379> JSON.DEL k1 .e[*]
(integer) 5
127.0.0.1:6379> JSON.GET k1
"{\"a\":{},\"b\":{\"a\":1},\"c\":{\"a\":1,\"b\":2},\"d\":{},\"e\":[]}"

```

JSON.FORGET

Sebuah alias dari [JSON.DEL](#)

JSON.GET

Kembalikan JSON serial pada satu atau beberapa jalur.

Sintaks

```

JSON.GET <key>
[INDENT indentation-string]
[NEWLINE newline-string]
[SPACE space-string]
[NOESCAPE]
[path ...]

```

- kunci (wajib) - Kunci Redis OSS dari jenis dokumen JSON
- INDENT/NEWLINE/SPACE (opsional) - mengontrol format string JSON yang dikembalikan, yaitu, “cetak cantik”. Nilai default masing-masing adalah string kosong. Mereka dapat diganti dalam kombinasi apa pun. Nilai dapat ditentukan dalam urutan apa pun.
- NOESCAPE - opsional, diizinkan hadir untuk kompatibilitas lama dan tidak memiliki efek lain.
- path (opsional) — nol atau lebih jalur JSON, default ke root jika tidak ada yang diberikan. Argumen jalur harus ditempatkan di akhir.

Nilai yang ditampilkan

Sintaksis jalur yang ditingkatkan:

Jika disediakan satu jalur:

- Kembalikan string serial dari array nilai.
- Jika tidak ada nilai yang dipilih, perintah ini mengembalikan array kosong.

Jika beberapa jalur diberikan:

- Kembalikan objek JSON dirangkai, di mana setiap jalur adalah kunci.
- Jika terdapat campuran sintaksis jalur yang ditingkatkan dan dibatasi, hasilnya sesuai dengan sintaksis yang ditingkatkan.
- Jika jalur tidak ada, nilai yang sesuai adalah array kosong.

Contoh

Sintaksis jalur yang ditingkatkan:

```
127.0.0.1:6379> JSON.SET k1 .
'{"firstName":"John","lastName":"Smith","age":27,"weight":135.25,"isAlive":true,"address":
{"street":"21 2nd Street","city":"New
York","state":"NY","zipcode":"10021-3100"},"phoneNumbers":
[{"type":"home","number":"212 555-1234"}, {"type":"office","number":"646
555-4567"}],"children":[],"spouse":null}'
OK
127.0.0.1:6379> JSON.GET k1 $.address.*
["\21 2nd Street","\New York","\NY","\10021-3100\"]
127.0.0.1:6379> JSON.GET k1 indent "\t" space " " NEWLINE "\n" $.address.*
["\n\t\21 2nd Street","\n\t\New York","\n\t\NY","\n\t\10021-3100\
\n"]
127.0.0.1:6379> JSON.GET k1 $.firstName $.lastName $.age
{"\$.firstName\":[\John\],"\$.lastName\":[\Smith\],"\$.age\":[27]}"
127.0.0.1:6379> JSON.SET k2 . '{"a":{ }, "b":{"a":1}, "c":{"a":1, "b":2}}'
OK
127.0.0.1:6379> json.get k2 $..*
"[{ },{\a\":1},{\a\":1,\b\":2},1,1,2]"
```

Sintaksis jalur terbatas:

```

127.0.0.1:6379> JSON.SET k1 .
'{"firstName":"John","lastName":"Smith","age":27,"weight":135.25,"isAlive":true,"address":
{"street":"21 2nd Street","city":"New
York","state":"NY","zipcode":"10021-3100"},"phoneNumbers":
[{"type":"home","number":"212 555-1234"}, {"type":"office","number":"646
555-4567"}],"children":[],"spouse":null}'
OK
127.0.0.1:6379> JSON.GET k1 .address
"{\"street\": \"21 2nd Street\", \"city\": \"New York\", \"state\": \"NY\", \"zipcode\":
\"10021-3100\"}"
127.0.0.1:6379> JSON.GET k1 indent "\t" space " " NEWLINE "\n" .address
"{\n\t\"street\": \"21 2nd Street\", \n\t\"city\": \"New York\", \n\t\"state\": \"NY\", \n
\t\"zipcode\": \"10021-3100\" \n}"
127.0.0.1:6379> JSON.GET k1 .firstName .lastName .age
"{\".firstName\": \"John\", \".lastName\": \"Smith\", \".age\": 27}"

```

JSON.MGET

Dapatkan JSON serial di jalur dari beberapa kunci dokumen. Kembalikan null untuk kunci yang tidak ada atau jalur JSON.

Sintaks

```
JSON.MGET <key> [key ...] <path>
```

- kunci (wajib) - Satu atau lebih kunci Redis OSS dari jenis dokumen.
- jalur (diperlukan) - jalur JSON

Nilai yang ditampilkan

- Array String Massal. Ukuran array sama dengan jumlah kunci dalam perintah. Setiap elemen array diisi dengan (a) JSON serial seperti yang terletak di jalur atau (b) Null jika kunci tidak ada atau jalur tidak ada dalam dokumen atau jalur tidak valid (kesalahan sintaks).
- Jika salah satu kunci yang ditentukan ada dan bukan kunci JSON, perintah ini mengembalikan kesalahan WRONGTYPE.

Contoh

Sintaksis jalur yang ditingkatkan:

```
127.0.0.1:6379> JSON.SET k1 . '{"address":{"street":"21 2nd Street","city":"New
York","state":"NY","zipcode":"10021"}}'
OK
127.0.0.1:6379> JSON.SET k2 . '{"address":{"street":"5 main
Street","city":"Boston","state":"MA","zipcode":"02101"}}'
OK
127.0.0.1:6379> JSON.SET k3 . '{"address":{"street":"100 Park
Ave","city":"Seattle","state":"WA","zipcode":"98102"}}'
OK
127.0.0.1:6379> JSON.MGET k1 k2 k3 $.address.city
1) "[\ "New York\ "]"
2) "[\ "Boston\ "]"
3) "[\ "Seattle\ "]"
```

Sintaksis jalur terbatas:

```
127.0.0.1:6379> JSON.SET k1 . '{"address":{"street":"21 2nd Street","city":"New
York","state":"NY","zipcode":"10021"}}'
OK
127.0.0.1:6379> JSON.SET k2 . '{"address":{"street":"5 main
Street","city":"Boston","state":"MA","zipcode":"02101"}}'
OK
127.0.0.1:6379> JSON.SET k3 . '{"address":{"street":"100 Park
Ave","city":"Seattle","state":"WA","zipcode":"98102"}}'
OK

127.0.0.1:6379> JSON.MGET k1 k2 k3 .address.city
1) "\"New York\""
2) "\"Seattle\""
3) "\"Seattle\""
```

JSON.NUMINCRBY

Naikkan nilai angka di jalur dengan nomor tertentu.

Sintaks

```
JSON.NUMINCRBY <key> <path> <number>
```

- kunci (wajib) - Kunci Redis OSS dari jenis dokumen JSON
- jalur (diperlukan) - jalur JSON
- nomor (wajib) — nomor

Nilai yang ditampilkan

Jika jalur adalah sintaksis yang ditingkatkan:

- Array String massal mewakili nilai yang dihasilkan di setiap jalur.
- Jika nilai bukan angka, nilai yang akan ditampilkan adalah kosong.
- Kesalahan `WRONGTYPE` jika angka tidak dapat diuraikan.
- Kesalahan `OVERFLOW` jika hasilnya di luar rentang double 64-bit IEEE.
- `NONEXISTENT` jika kunci dokumen tidak ada.

Jika jalur adalah sintaksis terbatas:

- String Massal mewakili nilai yang dihasilkan.
- Jika memilih beberapa nilai, perintah ini mengembalikan hasil nilai yang terakhir diperbarui.
- Kesalahan `WRONGTYPE` jika nilai di jalur bukan angka.
- Kesalahan `WRONGTYPE` jika angka tidak dapat diuraikan.
- Kesalahan `OVERFLOW` jika hasilnya di luar rentang double 64-bit IEEE.
- `NONEXISTENT` jika kunci dokumen tidak ada.

Contoh

Sintaksis jalur yang ditingkatkan:

```
127.0.0.1:6379> JSON.SET k1 . '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
OK
127.0.0.1:6379> JSON.NUMINCRBY k1 $.d[*] 10
"[11,12,13]"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[1],\"c\":[1,2],\"d\":[11,12,13]}"

127.0.0.1:6379> JSON.SET k1 $ '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
OK
127.0.0.1:6379> JSON.NUMINCRBY k1 $.a[*] 1
```

```

>[]"
127.0.0.1:6379> JSON.NUMINCRBY k1 $.b[*] 1
"[2]"
127.0.0.1:6379> JSON.NUMINCRBY k1 $.c[*] 1
"[2,3]"
127.0.0.1:6379> JSON.NUMINCRBY k1 $.d[*] 1
"[2,3,4]"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[2],\"c\":[2,3],\"d\":[2,3,4]}"

127.0.0.1:6379> JSON.SET k2 $ '{"a":{}, "b":{"a":1}, "c":{"a":1, "b":2}, "d":{"a":1, "b":2, "c":3}}'
OK
127.0.0.1:6379> JSON.NUMINCRBY k2 $.a.* 1
>[]"
127.0.0.1:6379> JSON.NUMINCRBY k2 $.b.* 1
"[2]"
127.0.0.1:6379> JSON.NUMINCRBY k2 $.c.* 1
"[2,3]"
127.0.0.1:6379> JSON.NUMINCRBY k2 $.d.* 1
"[2,3,4]"
127.0.0.1:6379> JSON.GET k2
"{\"a\":{\"a\":2},\"b\":{\"a\":2,\"b\":3},\"c\":{\"a\":2,\"b\":3,\"c\":4},\"d\":{\"a\":2,\"b\":3,\"c\":4},\"d\":{\"a\":2,\"b\":3,\"c\":4}}"

127.0.0.1:6379> JSON.SET k3 $ '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a", "b":"b"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.NUMINCRBY k3 $.a.* 1
"[null]"
127.0.0.1:6379> JSON.NUMINCRBY k3 $.b.* 1
"[null,2]"
127.0.0.1:6379> JSON.NUMINCRBY k3 $.c.* 1
"[null,null]"
127.0.0.1:6379> JSON.NUMINCRBY k3 $.d.* 1
"[2,null,4]"
127.0.0.1:6379> JSON.GET k3
"{\"a\":{\"a\":\"a\"},\"b\":{\"a\":\"a\", \"b\":2},\"c\":{\"a\":\"a\", \"b\":\"b\"},\"d\":{\"a\":2,\"b\":\"b\", \"c\":4},\"d\":{\"a\":2,\"b\":\"b\", \"c\":4}}"

```

Sintaksis jalur terbatas:

```
127.0.0.1:6379> JSON.SET k1 . '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
```

```

OK
127.0.0.1:6379> JSON.NUMINCRBY k1 .d[1] 10
"12"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[1],\"c\":[1,2],\"d\":[1,12,3]}"

127.0.0.1:6379> JSON.SET k1 . '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
OK
127.0.0.1:6379> JSON.NUMINCRBY k1 .a[*] 1
(error) NONEXISTENT JSON path does not exist
127.0.0.1:6379> JSON.NUMINCRBY k1 .b[*] 1
"2"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[2],\"c\":[1,2],\"d\":[1,2,3]}"
127.0.0.1:6379> JSON.NUMINCRBY k1 .c[*] 1
"3"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[2],\"c\":[2,3],\"d\":[1,2,3]}"
127.0.0.1:6379> JSON.NUMINCRBY k1 .d[*] 1
"4"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[2],\"c\":[2,3],\"d\":[2,3,4]}"

127.0.0.1:6379> JSON.SET k2 . '{"a":{}, "b":{"a":1}, "c":{"a":1, "b":2}, "d":{"a":1, "b":2, "c":3}}'
OK
127.0.0.1:6379> JSON.NUMINCRBY k2 .a.* 1
(error) NONEXISTENT JSON path does not exist
127.0.0.1:6379> JSON.NUMINCRBY k2 .b.* 1
"2"
127.0.0.1:6379> JSON.GET k2
"{\"a\":{},\"b\":{\"a\":2},\"c\":{\"a\":1,\"b\":2},\"d\":{\"a\":1,\"b\":2,\"c\":3}}"
127.0.0.1:6379> JSON.NUMINCRBY k2 .c.* 1
"3"
127.0.0.1:6379> JSON.GET k2
"{\"a\":{},\"b\":{\"a\":2},\"c\":{\"a\":2,\"b\":3},\"d\":{\"a\":1,\"b\":2,\"c\":3}}"
127.0.0.1:6379> JSON.NUMINCRBY k2 .d.* 1
"4"
127.0.0.1:6379> JSON.GET k2
"{\"a\":{},\"b\":{\"a\":2},\"c\":{\"a\":2,\"b\":3},\"d\":{\"a\":2,\"b\":3,\"c\":4}}"

127.0.0.1:6379> JSON.SET k3 . '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a", "b":"b"}, "d":{"a":1, "b":"b", "c":3}}'
OK

```

```
127.0.0.1:6379> JSON.NUMINCRBY k3 .a.* 1
(error) WRONGTYPE JSON element is not a number
127.0.0.1:6379> JSON.NUMINCRBY k3 .b.* 1
"2"
127.0.0.1:6379> JSON.NUMINCRBY k3 .c.* 1
(error) WRONGTYPE JSON element is not a number
127.0.0.1:6379> JSON.NUMINCRBY k3 .d.* 1
"4"
```

JSON.NUMMULTBY

Lipat gandakan nilai angka di jalur dengan angka tertentu.

Sintaks

```
JSON.NUMMULTBY <key> <path> <number>
```

- kunci (wajib) - Kunci Redis OSS dari jenis dokumen JSON
- jalur (diperlukan) - jalur JSON
- nomor (wajib) — nomor

Nilai yang ditampilkan

Jika jalur adalah sintaksis yang ditingkatkan:

- Array String massal mewakili nilai yang dihasilkan di setiap jalur.
- Jika nilai bukan angka, nilai yang akan ditampilkan adalah kosong.
- Kesalahan `WRONGTYPE` jika angka tidak dapat diuraikan.
- Kesalahan `OVERFLOW` jika hasilnya di luar rentang double 64-bit IEEE.
- `NONEXISTENT` jika kunci dokumen tidak ada.

Jika jalur adalah sintaksis terbatas:

- String Massal mewakili nilai yang dihasilkan.
- Jika memilih beberapa nilai, perintah ini mengembalikan hasil nilai yang terakhir diperbarui.
- Kesalahan `WRONGTYPE` jika nilai di jalur bukan angka.

- Kesalahan `WRONGTYPE` jika angka tidak dapat diuraikan.
- Kesalahan `OVERFLOW` jika hasilnya di luar rentang double 64-bit IEEE.
- `NONEXISTENT` jika kunci dokumen tidak ada.

Contoh

Sintaksis jalur yang ditingkatkan:

```
127.0.0.1:6379> JSON.SET k1 . '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k1 $.d[*] 2
"[2,4,6]"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[1],\"c\":[1,2],\"d\":[2,4,6]}"

127.0.0.1:6379> JSON.SET k1 $ '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k1 $.a[*] 2
"[]"
127.0.0.1:6379> JSON.NUMMULTBY k1 $.b[*] 2
"[2]"
127.0.0.1:6379> JSON.NUMMULTBY k1 $.c[*] 2
"[2,4]"
127.0.0.1:6379> JSON.NUMMULTBY k1 $.d[*] 2
"[2,4,6]"

127.0.0.1:6379> JSON.SET k2 $ '{"a":{}, "b":{"a":1}, "c":{"a":1, "b":2}, "d":{"a":1, "b":2, "c":3}}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k2 $.a.* 2
"[]"
127.0.0.1:6379> JSON.NUMMULTBY k2 $.b.* 2
"[2]"
127.0.0.1:6379> JSON.NUMMULTBY k2 $.c.* 2
"[2,4]"
127.0.0.1:6379> JSON.NUMMULTBY k2 $.d.* 2
"[2,4,6]"

127.0.0.1:6379> JSON.SET k3 $ '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a", "b":"b"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k3 $.a.* 2
```



```

"[null]"
127.0.0.1:6379> JSON.NUMMULTBY k3 $.b.* 2
"[null,2]"
127.0.0.1:6379> JSON.NUMMULTBY k3 $.c.* 2
"[null,null]"
127.0.0.1:6379> JSON.NUMMULTBY k3 $.d.* 2
"[2,null,6]"

```

Sintaksis jalur terbatas:

```

127.0.0.1:6379> JSON.SET k1 . '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k1 .d[1] 2
"4"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[1],\"c\":[1,2],\"d\":[1,4,3]}"

127.0.0.1:6379> JSON.SET k1 . '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k1 .a[*] 2
(error) NONEXISTENT JSON path does not exist
127.0.0.1:6379> JSON.NUMMULTBY k1 .b[*] 2
"2"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[2],\"c\":[1,2],\"d\":[1,2,3]}"
127.0.0.1:6379> JSON.NUMMULTBY k1 .c[*] 2
"4"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[2],\"c\":[2,4],\"d\":[1,2,3]}"
127.0.0.1:6379> JSON.NUMMULTBY k1 .d[*] 2
"6"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[2],\"c\":[2,4],\"d\":[2,4,6]}"

127.0.0.1:6379> JSON.SET k2 . '{"a":{}, "b":{"a":1}, "c":{"a":1, "b":2}, "d":{"a":1, "b":2, "c":3}}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k2 .a.* 2
(error) NONEXISTENT JSON path does not exist
127.0.0.1:6379> JSON.NUMMULTBY k2 .b.* 2
"2"
127.0.0.1:6379> JSON.GET k2

```

```

"{\"a\":{},\"b\":{\"a\":2},\"c\":{\"a\":1,\"b\":2},\"d\":{\"a\":1,\"b\":2,\"c\":3}}"
127.0.0.1:6379> JSON.NUMMULTBY k2 .c.* 2
"4"
127.0.0.1:6379> JSON.GET k2
"{\"a\":{},\"b\":{\"a\":2},\"c\":{\"a\":2,\"b\":4},\"d\":{\"a\":1,\"b\":2,\"c\":3}}"
127.0.0.1:6379> JSON.NUMMULTBY k2 .d.* 2
"6"
127.0.0.1:6379> JSON.GET k2
"{\"a\":{},\"b\":{\"a\":2},\"c\":{\"a\":2,\"b\":4},\"d\":{\"a\":2,\"b\":4,\"c\":6}}"

127.0.0.1:6379> JSON.SET k3 . '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a",
  "b":"b"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k3 .a.* 2
(error) WRONGTYPE JSON element is not a number
127.0.0.1:6379> JSON.NUMMULTBY k3 .b.* 2
"2"
127.0.0.1:6379> JSON.GET k3
"{\"a\":{\"a\":\"a\"},\"b\":{\"a\":\"a\", \"b\":2},\"c\":{\"a\":\"a\", \"b\":\"b\"},\"d
\":{ \"a\":1, \"b\":\"b\", \"c\":3}}"
127.0.0.1:6379> JSON.NUMMULTBY k3 .c.* 2
(error) WRONGTYPE JSON element is not a number
127.0.0.1:6379> JSON.NUMMULTBY k3 .d.* 2
"6"
127.0.0.1:6379> JSON.GET k3
"{\"a\":{\"a\":\"a\"},\"b\":{\"a\":\"a\", \"b\":2},\"c\":{\"a\":\"a\", \"b\":\"b\"},\"d
\":{ \"a\":2, \"b\":\"b\", \"c\":6}}"

```

JSON.OBJLEN

Dapatkan jumlah kunci dalam nilai objek di jalur.

Sintaks

```
JSON.OBJLEN <key> [path]
```

- kunci (wajib) - Kunci Redis OSS dari jenis dokumen JSON
- path (opsional) - jalur JSON. Default ke root jika tidak disediakan

Nilai yang ditampilkan

Jika jalur adalah sintaksis yang ditingkatkan:

- Array bilangan bulat, mewakili panjang objek di setiap jalur.
- Jika nilai bukan objek, nilai yang akan dikembalikan adalah nilai yang kosong.
- Kosong jika kunci dokumen tidak ada.

Jika jalur adalah sintaksis terbatas:

- Integer, jumlah kunci dalam objek.
- Jika memilih beberapa objek, perintah ini mengembalikan panjang objek pertama.
- Kesalahan `WRONGTYPE` jika nilai di jalur bukan objek.
- Kesalahan `WRONGTYPE` jika jalur tidak ada.
- Kosong jika kunci dokumen tidak ada.

Contoh

Sintaksis jalur yang ditingkatkan:

```
127.0.0.1:6379> JSON.SET k1 $ '{"a":{}, "b":{"a":"a"}, "c":{"a":"a", "b":"bb"}, "d":
{"a":1, "b":"b", "c":{"a":3,"b":4}}, "e":1}'
OK
127.0.0.1:6379> JSON.OBJLEN k1 $.a
1) (integer) 0
127.0.0.1:6379> JSON.OBJLEN k1 $.a.*
(empty array)
127.0.0.1:6379> JSON.OBJLEN k1 $.b
1) (integer) 1
127.0.0.1:6379> JSON.OBJLEN k1 $.b.*
1) (nil)
127.0.0.1:6379> JSON.OBJLEN k1 $.c
1) (integer) 2
127.0.0.1:6379> JSON.OBJLEN k1 $.c.*
1) (nil)
2) (nil)
127.0.0.1:6379> JSON.OBJLEN k1 $.d
1) (integer) 3
127.0.0.1:6379> JSON.OBJLEN k1 $.d.*
1) (nil)
2) (nil)
```

```

3) (integer) 2
127.0.0.1:6379> JSON.OBJLEN k1 $.*
1) (integer) 0
2) (integer) 1
3) (integer) 2
4) (integer) 3
5) (nil)

```

Sintaksis jalur terbatas:

```

127.0.0.1:6379> JSON.SET k1 . '{"a":{}, "b":{"a":"a"}, "c":{"a":"a", "b":"bb"}, "d":
{"a":1, "b":"b", "c":{"a":3,"b":4}}, "e":1}'
OK
127.0.0.1:6379> JSON.OBJLEN k1 .a
(integer) 0
127.0.0.1:6379> JSON.OBJLEN k1 .a.*
(error) NONEXISTENT JSON path does not exist
127.0.0.1:6379> JSON.OBJLEN k1 .b
(integer) 1
127.0.0.1:6379> JSON.OBJLEN k1 .b.*
(error) WRONGTYPE JSON element is not an object
127.0.0.1:6379> JSON.OBJLEN k1 .c
(integer) 2
127.0.0.1:6379> JSON.OBJLEN k1 .c.*
(error) WRONGTYPE JSON element is not an object
127.0.0.1:6379> JSON.OBJLEN k1 .d
(integer) 3
127.0.0.1:6379> JSON.OBJLEN k1 .d.*
(integer) 2
127.0.0.1:6379> JSON.OBJLEN k1 .*
(integer) 0

```

JSON.OBJKEYS

Dapatkan nama kunci dalam nilai objek di jalur.

Sintaks

```
JSON.OBJKEYS <key> [path]
```

- kunci (wajib) - Kunci Redis OSS dari jenis dokumen JSON
- path (opsional) - jalur JSON. Default ke root jika tidak disediakan

Nilai yang ditampilkan

Jika jalur adalah sintaksis yang ditingkatkan:

- Array string massal. Setiap elemen adalah array kunci dalam objek yang cocok.
- Jika nilai bukan objek, nilai yang akan dikembalikan adalah nilai yang kosong.
- Kosong jika kunci dokumen tidak ada.

Jika jalur adalah sintaksis terbatas:

- Array string massal. Setiap elemen adalah nama kunci dalam objek.
- Jika memilih beberapa objek, perintah ini mengembalikan kunci objek pertama.
- Kesalahan `WRONGTYPE` jika nilai di jalur bukan objek.
- Kesalahan `WRONGTYPE` jika jalur tidak ada.
- Kosong jika kunci dokumen tidak ada.

Contoh

Sintaksis jalur yang ditingkatkan:

```
127.0.0.1:6379> JSON.SET k1 $ '{"a":{}, "b":{"a":"a"}, "c":{"a":"a", "b":"bb"}, "d":
{"a":1, "b":"b", "c":{"a":3, "b":4}}, "e":1}'
OK
127.0.0.1:6379> JSON.OBJKEYS k1 $.*
1) (empty array)
2) 1) "a"
3) 1) "a"
   2) "b"
4) 1) "a"
   2) "b"
   3) "c"
5) (empty array)
127.0.0.1:6379> JSON.OBJKEYS k1 $.d
1) 1) "a"
   2) "b"
```

```
3) "c"
```

Sintaksis jalur terbatas:

```
127.0.0.1:6379> JSON.SET k1 $ '{"a":{}, "b":{"a":"a"}, "c":{"a":"a", "b":"bb"}, "d":
{"a":1, "b":"b", "c":{"a":3, "b":4}}, "e":1}'
OK
127.0.0.1:6379> JSON.OBJKEYS k1 .*
1) "a"
127.0.0.1:6379> JSON.OBJKEYS k1 .d
1) "a"
2) "b"
3) "c"
```

JSON.RESP

Kembalikan nilai JSON di jalur yang diberikan di Redis OSS Serialization Protocol (RESP). Jika nilainya adalah wadah, responsnya adalah array RESP atau array bersarang.

- JSON kosong dipetakan ke String Massal Kosong RESP.
- Nilai boolean JSON dipetakan ke masing-masing Resp Simple Strings.
- Angka integer dipetakan ke Integer RESP.
- Nomor floating point ganda IEEE 64-bit dipetakan ke String Massal RESP.
- String JSON dipetakan ke Resp Bulk Strings.
- Array JSON direpresentasikan sebagai Array RESP, di mana elemen pertama adalah string sederhana [, diikuti oleh elemen array.
- Objek JSON direpresentasikan sebagai Array RESP, di mana elemen pertama adalah string sederhana {, diikuti oleh pasangan kunci-nilai, yang masing-masing adalah string massal RESP.

Sintaks

```
JSON.RESP <key> [path]
```

- kunci (wajib) - Kunci Redis OSS dari jenis dokumen JSON
- path (opsional) - jalur JSON. Default ke root jika tidak disediakan

Nilai yang ditampilkan

Jika jalur adalah sintaksis yang ditingkatkan:

- Array dari array. Setiap elemen array merepresentasikan bentuk RESP dari nilai pada satu jalur.
- Array kosong jika kunci dokumen tidak ada.

Jika jalur adalah sintaksis terbatas:

- Array, mewakili bentuk RESP dari nilai di jalur.
- Kosong jika kunci dokumen tidak ada.

Contoh

Sintaksis jalur yang ditingkatkan:

```
127.0.0.1:6379> JSON.SET k1 .
'{"firstName":"John","lastName":"Smith","age":27,"weight":135.25,"isAlive":true,"address":
{"street":"21 2nd Street","city":"New
York","state":"NY","zipcode":"10021-3100"},"phoneNumbers":
[{"type":"home","number":"212 555-1234"},{"type":"office","number":"646
555-4567"}],"children":[],"spouse":null}'
OK
```

```
127.0.0.1:6379> JSON.RESP k1 $.address
```

```
1) 1) {
  2) 1) "street"
     2) "21 2nd Street"
  3) 1) "city"
     2) "New York"
  4) 1) "state"
     2) "NY"
  5) 1) "zipcode"
     2) "10021-3100"
```

```
127.0.0.1:6379> JSON.RESP k1 $.address.*
```

```
1) "21 2nd Street"
2) "New York"
3) "NY"
4) "10021-3100"
```

```
127.0.0.1:6379> JSON.RESP k1 $.phoneNumbers
1) 1) [
  2) 1) {
    2) 1) "type"
    2) "home"
    3) 1) "number"
    2) "555 555-1234"
  3) 1) {
    2) 1) "type"
    2) "office"
    3) 1) "number"
    2) "555 555-4567"

127.0.0.1:6379> JSON.RESP k1 $.phoneNumbers[*]
1) 1) {
  2) 1) "type"
  2) "home"
  3) 1) "number"
  2) "212 555-1234"
2) 1) {
  2) 1) "type"
  2) "office"
  3) 1) "number"
  2) "555 555-4567"
```

Sintaksis jalur terbatas:

```
127.0.0.1:6379> JSON.SET k1 .
'{"firstName":"John","lastName":"Smith","age":27,"weight":135.25,"isAlive":true,"address":
{"street":"21 2nd Street","city":"New
York","state":"NY","zipcode":"10021-3100"},"phoneNumbers":
[{"type":"home","number":"212 555-1234"},{"type":"office","number":"646
555-4567"}],"children":[],"spouse":null}'
OK

127.0.0.1:6379> JSON.RESP k1 .address
1) {
2) 1) "street"
  2) "21 2nd Street"
3) 1) "city"
  2) "New York"
4) 1) "state"
```



```
2) "NY"
5) 1) "zipcode"
   2) "10021-3100"

127.0.0.1:6379> JSON.RESP k1
1) {
2) 1) "firstName"
   2) "John"
3) 1) "lastName"
   2) "Smith"
4) 1) "age"
   2) (integer) 27
5) 1) "weight"
   2) "135.25"
6) 1) "isAlive"
   2) true
7) 1) "address"
   2) 1) {
      2) 1) "street"
         2) "21 2nd Street"
      3) 1) "city"
         2) "New York"
      4) 1) "state"
         2) "NY"
      5) 1) "zipcode"
         2) "10021-3100"
8) 1) "phoneNumbers"
   2) 1) [
      2) 1) {
         2) 1) "type"
            2) "home"
         3) 1) "number"
            2) "212 555-1234"
      3) 1) {
         2) 1) "type"
            2) "office"
         3) 1) "number"
            2) "555 555-4567"
9) 1) "children"
   2) 1) [
10) 1) "spouse"
     2) (nil)
```

JSON.SET

Tetapkan nilai JSON di jalur.

Jika jalur memanggil anggota objek:

- Jika elemen induk tidak ada, perintah akan mengembalikan kesalahan NONEXISTENT.
- Jika elemen induk ada tetapi bukan objek, perintah akan mengembalikan ERROR.
- Jika elemen induk ada dan merupakan objek:
 - Jika anggota tidak ada, anggota baru akan ditambahkan ke objek induk jika dan hanya jika objek induk adalah turunan terakhir di jalur. Jika tidak, perintah akan mengembalikan kesalahan NONEXISTENT.
 - Jika anggota ada, nilainya akan diganti dengan nilai JSON.

Jika jalur memanggil indeks array:

- Jika elemen induk tidak ada, perintah akan mengembalikan kesalahan NONEXISTENT.
- Jika elemen induk ada tetapi bukan array, perintah akan mengembalikan ERROR.
- Jika elemen induk ada tetapi indeks di luar batas, perintah akan mengembalikan kesalahan OUTFBORDIES.
- Jika elemen induk ada dan indeks valid, elemen akan diganti dengan nilai JSON baru.

Jika jalur memanggil objek atau array, nilai (objek atau array) akan digantikan oleh nilai JSON baru.

Sintaks

```
JSON.SET <key> <path> <json> [NX | XX]
```

[NX | XX] Di mana Anda dapat memiliki 0 atau 1 pengidentifikasi [NX | XX]

- kunci (wajib) - Kunci Redis OSS dari jenis dokumen JSON
- jalur (diperlukan) - jalur JSON. Untuk kunci Redis OSS baru, jalur JSON harus menjadi root “.”.
- NX (opsional) - Jika jalurnya adalah root, atur nilainya hanya jika kunci Redis OSS tidak ada, yaitu menyisipkan dokumen baru. Jika jalur bukan root, atur nilainya hanya jika jalur tidak ada, yaitu, masukkan nilai ke dalam dokumen.

- **XX** (opsional) - Jika jalurnya adalah root, atur nilainya hanya jika kunci Redis OSS ada, yaitu mengganti dokumen yang ada. Jika jalur bukan root, tetapkan nilainya hanya jika jalur ada, yaitu, perbarui nilai yang ada.

Nilai yang ditampilkan

- String sederhana 'OK' jika berhasil.
- Kosong jika kondisi NX atau XX tidak terpenuhi.

Contoh

Sintaksis jalur yang ditingkatkan:

```
127.0.0.1:6379> JSON.SET k1 . '{"a":{"a":1, "b":2, "c":3}}'
OK
127.0.0.1:6379> JSON.SET k1 $.a.* '0'
OK
127.0.0.1:6379> JSON.GET k1
"{\"a\":{\"a\":0,\"b\":0,\"c\":0}}"

127.0.0.1:6379> JSON.SET k2 . '{"a": [1,2,3,4,5]}'
OK
127.0.0.1:6379> JSON.SET k2 $.a[*] '0'
OK
127.0.0.1:6379> JSON.GET k2
"{\"a\":[0,0,0,0,0]}"
```

Sintaksis jalur terbatas:

```
127.0.0.1:6379> JSON.SET k1 . '{"c":{"a":1, "b":2}, "e": [1,2,3,4,5]}'
OK
127.0.0.1:6379> JSON.SET k1 .c.a '0'
OK
127.0.0.1:6379> JSON.GET k1
"{\"c\":{\"a\":0,\"b\":2},\"e\":[1,2,3,4,5]}"
127.0.0.1:6379> JSON.SET k1 .e[-1] '0'
OK
127.0.0.1:6379> JSON.GET k1
"{\"c\":{\"a\":0,\"b\":2},\"e\":[1,2,3,4,0]}"
127.0.0.1:6379> JSON.SET k1 .e[5] '0'
```

```
(error) OUTOFBOUNDARIES Array index is out of bounds
```

JSON.STRAPPEND

Tambahkan string ke string JSON di jalur.

Sintaks

```
JSON.STRAPPEND <key> [path] <json_string>
```

- kunci (wajib) - Kunci Redis OSS dari jenis dokumen JSON
- path (opsional) - jalur JSON. Default ke root jika tidak disediakan
- json_string (wajib) - representasi JSON dari string. Perhatikan bahwa string JSON harus dikutip, yaitu, "foo".

Nilai yang ditampilkan

Jika jalur adalah sintaksis yang ditingkatkan:

- Array bilangan bulat, mewakili panjang baru string di setiap jalur.
- Jika nilai di jalur bukan string, nilai yang akan dikembalikan adalah kosong.
- SYNTAXERR kesalahan jika argumen input json bukan string JSON yang valid.
- NONEXISTENT kesalahan jika jalur tidak ada.

Jika jalur adalah sintaksis terbatas:

- Integer, panjang baru string.
- Jika memilih beberapa nilai string, perintah ini mengembalikan panjang baru dari string yang terakhir diperbarui.
- Kesalahan WRONGTYPE jika nilai di jalur bukan string.
- Kesalahan WRONGTYPE jika argumen input json bukan string JSON yang valid.
- Kesalahan NONEXISTENT jika jalur tidak ada.

Contoh

Sintaksis jalur yang ditingkatkan:

```

127.0.0.1:6379> JSON.SET k1 $ '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a",
"b":"bb"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.STRAPPEND k1 $.a.a 'a'
1) (integer) 2
127.0.0.1:6379> JSON.STRAPPEND k1 $.a.* 'a'
1) (integer) 3
127.0.0.1:6379> JSON.STRAPPEND k1 $.b.* 'a'
1) (integer) 2
2) (nil)
127.0.0.1:6379> JSON.STRAPPEND k1 $.c.* 'a'
1) (integer) 2
2) (integer) 3
127.0.0.1:6379> JSON.STRAPPEND k1 $.c.b 'a'
1) (integer) 4
127.0.0.1:6379> JSON.STRAPPEND k1 $.d.* 'a'
1) (nil)
2) (integer) 2
3) (nil)

```

Sintaksis jalur terbatas:

```

127.0.0.1:6379> JSON.SET k1 . '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a",
"b":"bb"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.STRAPPEND k1 .a.a 'a'
(integer) 2
127.0.0.1:6379> JSON.STRAPPEND k1 .a.* 'a'
(integer) 3
127.0.0.1:6379> JSON.STRAPPEND k1 .b.* 'a'
(integer) 2
127.0.0.1:6379> JSON.STRAPPEND k1 .c.* 'a'
(integer) 3
127.0.0.1:6379> JSON.STRAPPEND k1 .c.b 'a'
(integer) 4
127.0.0.1:6379> JSON.STRAPPEND k1 .d.* 'a'
(integer) 2

```

JSON.STRLEN

Dapatkan panjang nilai string JSON di jalur.

Sintaks

```
JSON.STRLEN <key> [path]
```

- kunci (wajib) - Kunci Redis OSS dari jenis dokumen JSON
- path (opsional) - jalur JSON. Default ke root jika tidak disediakan

Nilai yang ditampilkan

Jika jalur adalah sintaksis yang ditingkatkan:

- Array bilangan bulat, mewakili panjang nilai string di setiap jalur.
- Jika nilai bukan string, nilai yang akan dikembalikan adalah kosong.
- Kosong jika kunci dokumen tidak ada.

Jika jalur adalah sintaksis terbatas:

- Integer, panjang string.
- Jika memilih beberapa nilai string, perintah ini mengembalikan panjang string pertama.
- Kesalahan `WRONGTYPE` jika nilai di jalur bukan string.
- Kesalahan `NONEXISTENT` jika jalur tidak ada.
- Kosong jika kunci dokumen tidak ada.

Contoh

Sintaksis jalur yang ditingkatkan:

```
127.0.0.1:6379> JSON.SET k1 $ '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a", "b":"bb"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.STRLEN k1 $.a.a
1) (integer) 1
127.0.0.1:6379> JSON.STRLEN k1 $.a.*
```

```

1) (integer) 1
127.0.0.1:6379> JSON.STRLEN k1 $.c.*
1) (integer) 1
2) (integer) 2
127.0.0.1:6379> JSON.STRLEN k1 $.c.b
1) (integer) 2
127.0.0.1:6379> JSON.STRLEN k1 $.d.*
1) (nil)
2) (integer) 1
3) (nil)

```

Sintaksis jalur terbatas:

```

127.0.0.1:6379> JSON.SET k1 $ '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a",
"b":"bb"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.STRLEN k1 .a.a
(integer) 1
127.0.0.1:6379> JSON.STRLEN k1 .a.*
(integer) 1
127.0.0.1:6379> JSON.STRLEN k1 .c.*
(integer) 1
127.0.0.1:6379> JSON.STRLEN k1 .c.b
(integer) 2
127.0.0.1:6379> JSON.STRLEN k1 .d.*
(integer) 1

```

JSON.TOGGLE

Alihkan nilai boolean antara true dan false di jalur.

Sintaks

```
JSON.TOGGLE <key> [path]
```

- kunci (wajib) - Kunci Redis OSS dari jenis dokumen JSON
- path (opsional) - jalur JSON. Default ke root jika tidak disediakan

Nilai yang ditampilkan

Jika jalur adalah sintaksis yang ditingkatkan:

- Array bilangan bulat (0 - false, 1 - true) mewakili nilai boolean yang dihasilkan di setiap jalur.
- Jika nilai bukan boolean, nilai kembalinya yang sesuai adalah null.
- NONEXISTENT jika kunci dokumen tidak ada.

Jika jalur adalah sintaksis terbatas:

- String ("true" /"false") mewakili nilai boolean yang dihasilkan.
- NONEXISTENT jika kunci dokumen tidak ada.
- WRONGTYPEkesalahan jika nilai di jalur bukan boolean.

Contoh

Sintaksis jalur yang ditingkatkan:

```
127.0.0.1:6379> JSON.SET k1 . '{"a":true, "b":false, "c":1, "d":null, "e":"foo", "f":
[], "g":{}}'
OK
127.0.0.1:6379> JSON.TOGGLE k1 $.*
1) (integer) 0
2) (integer) 1
3) (nil)
4) (nil)
5) (nil)
6) (nil)
7) (nil)
127.0.0.1:6379> JSON.TOGGLE k1 $.*
1) (integer) 1
2) (integer) 0
3) (nil)
4) (nil)
5) (nil)
6) (nil)
7) (nil)
```

Sintaksis jalur terbatas:

```
127.0.0.1:6379> JSON.SET k1 . true
```



```
OK
127.0.0.1:6379> JSON.TOGGLE k1
"false"
127.0.0.1:6379> JSON.TOGGLE k1
"true"

127.0.0.1:6379> JSON.SET k2 . '{"isAvailable": false}'
OK
127.0.0.1:6379> JSON.TOGGLE k2 .isAvailable
"true"
127.0.0.1:6379> JSON.TOGGLE k2 .isAvailable
"false"
```

JSON.TYPE

Laporkan jenis nilai pada jalur yang diberikan.

Sintaks

```
JSON.TYPE <key> [path]
```

- kunci (wajib) - Kunci Redis OSS dari jenis dokumen JSON
- path (opsional) - jalur JSON. Default ke root jika tidak disediakan

Nilai yang ditampilkan

Jika jalur adalah sintaksis yang ditingkatkan:

- Array string, mewakili jenis nilai di setiap jalur. Jenisnya adalah salah satu dari {"null", "boolean", "string", "number", "integer", "object" dan "array"}.
- Jika jalur tidak ada, nilai yang ditampilkan adalah kosong.
- Array kosong jika kunci dokumen tidak ada.

Jika jalur adalah sintaksis terbatas:

- String, jenis nilai
- Kosong jika kunci dokumen tidak ada.

- Kosong jika jalur JSON tidak valid atau tidak ada.

Contoh

Sintaksis jalur yang ditingkatkan:

```
127.0.0.1:6379> JSON.SET k1 . '[1, 2.3, "foo", true, null, {}, []]'
OK
127.0.0.1:6379> JSON.TYPE k1 $[*]
1) integer
2) number
3) string
4) boolean
5) null
6) object
7) array
```

Sintaksis jalur terbatas:

```
127.0.0.1:6379> JSON.SET k1 .
'{"firstName":"John","lastName":"Smith","age":27,"weight":135.25,"isAlive":true,"address":
{"street":"21 2nd Street","city":"New
York","state":"NY","zipcode":"10021-3100"},"phoneNumbers":
[{"type":"home","number":"212 555-1234"}, {"type":"office","number":"646
555-4567"}],"children":[],"spouse":null}'
OK
127.0.0.1:6379> JSON.TYPE k1
object
127.0.0.1:6379> JSON.TYPE k1 .children
array
127.0.0.1:6379> JSON.TYPE k1 .firstName
string
127.0.0.1:6379> JSON.TYPE k1 .age
integer
127.0.0.1:6379> JSON.TYPE k1 .weight
number
127.0.0.1:6379> JSON.TYPE k1 .isAlive
boolean
127.0.0.1:6379> JSON.TYPE k1 .spouse
null
```

Menandai sumber daya MemoryDB Anda

Untuk membantu Anda mengelola cluster dan sumber daya MemoryDB lainnya, Anda dapat menetapkan metadata Anda sendiri ke setiap sumber daya dalam bentuk tag. Tag memungkinkan Anda untuk mengkategorikan AWS sumber daya Anda dengan cara yang berbeda, misalnya, berdasarkan tujuan, pemilik, atau lingkungan. Hal ini berguna ketika Anda memiliki banyak sumber daya dengan jenis yang sama—Anda dapat dengan cepat mengidentifikasi sumber daya tertentu berdasarkan tag yang telah Anda tetapkan. Topik ini menjelaskan tag dan menunjukkan cara membuatnya.

Warning

Sebagai praktik terbaik, sebaiknya Anda tidak menyertakan data sensitif ke dalam tag.

Dasar-dasar tag

Tag adalah label yang Anda tetapkan ke AWS sumber daya. Setiap tag terdiri dari kunci dan nilai opsional, yang keduanya Anda tentukan. Tag memungkinkan Anda untuk mengkategorikan AWS sumber daya Anda dengan cara yang berbeda, misalnya, berdasarkan tujuan atau pemilik. Misalnya, Anda dapat menentukan satu set tag untuk kluster MemoryDB akun Anda yang membantu Anda melacak pemilik dan grup pengguna masing-masing cluster.

Sebaiknya rancang serangkaian kunci tag yang memenuhi kebutuhan setiap jenis sumber daya. Penggunaan set kunci tag yang konsisten akan memudahkan manajemen sumber daya Anda. Anda dapat mencari dan memfilter sumber daya berdasarkan tag yang Anda tambahkan. Untuk informasi selengkapnya tentang cara mengimplementasikan strategi pemberian tag sumber daya yang efektif, lihat [Laporan resmi AWS Praktik Terbaik Pemberian Tag](#).

Tag tidak memiliki arti semantik untuk MemoryDB dan ditafsirkan secara ketat sebagai serangkaian karakter. Selain itu, tag tidak secara otomatis ditetapkan ke sumber daya Anda. Anda dapat mengedit kunci dan nilai tag, serta menghapus tag dari sumber daya kapan saja. Anda dapat menetapkan nilai tag ke null. Jika Anda menambahkan tag yang memiliki kunci yang sama dengan tag yang sudah ada di sumber daya tersebut, nilai yang baru akan menimpa nilai yang lama. Jika sumber daya dihapus, semua tanda untuk sumber daya tersebut juga akan dihapus.

Anda dapat bekerja dengan tag menggunakan AWS Management Console, the AWS CLI, dan MemoryDB API.

Jika Anda menggunakan IAM, Anda dapat mengontrol pengguna mana di AWS akun Anda yang memiliki izin untuk membuat, mengedit, atau menghapus tag. Untuk informasi selengkapnya, lihat [Izin tingkat sumber daya](#).

Sumber daya yang dapat Anda beri tag

Anda dapat menandai sebagian besar sumber daya MemoryDB yang sudah ada di akun Anda. Tabel di bawah ini mencantumkan sumber daya yang mendukung pemberian tag. Jika Anda menggunakan AWS Management Console, Anda dapat menerapkan tag ke sumber daya dengan menggunakan [Editor Tag](#). Beberapa layar sumber daya memungkinkan Anda menentukan tag untuk sebuah sumber daya saat sumber daya tersebut dibuat; misalnya, tag dengan kunci Nama dan nilai yang Anda tentukan. Dalam kebanyakan kasus, konsol menerapkan tag segera setelah sumber daya dibuat (bukan selama pembuatan sumber daya). Konsol dapat mengatur sumber daya sesuai dengan tag Nama, tetapi tag ini tidak memiliki arti semantik untuk layanan MemoryDB.

Selain itu, beberapa tindakan pembuatan sumber daya memungkinkan Anda menentukan tag untuk sumber daya saat sumber daya tersebut dibuat. Jika tag tidak dapat diterapkan selama pembuatan sumber daya, kami akan mengembalikan proses pembuatan sumber daya. Hal ini untuk memastikan bahwa sumber daya dibuat dengan tag atau tidak akan dibuat sama sekali, dan tidak akan ada sumber daya yang dibiarkan tidak bertanda. Dengan memberikan tag pada sumber daya saat pembuatan, Anda tidak perlu menjalankan skrip pemberian tag kustom setelah pembuatan sumber daya.

Jika Anda menggunakan Amazon MemoryDB API, AWS CLI, atau AWS SDK, Anda dapat menggunakan Tags parameter pada tindakan API MemoryDB yang relevan untuk menerapkan tag. File tersebut adalah:

- `CreateCluster`
- `CopySnapshot`
- `CreateParameterGroup`
- `CreateSubnetGroup`
- `CreateSnapshot`
- `CreateACL`
- `CreateUser`

Tabel berikut menjelaskan sumber daya MemoryDB yang dapat ditandai, dan sumber daya yang dapat ditandai pada pembuatan menggunakan API MemoryDB, AWS CLI, atau SDK. AWS

Menandai dukungan untuk sumber daya MemoryDB

Mendukung tag	Mendukung penandaan saat pembuatan
Ya	Ya
Ya	Ya
Ya	Ya
Ya	Ya
Ya	Ya
Ya	Ya

Anda dapat menerapkan izin tingkat sumber daya berbasis tag dalam kebijakan IAM Anda ke tindakan API MemoryDB yang mendukung penandaan pada pembuatan untuk menerapkan kontrol terperinci atas pengguna dan grup yang dapat menandai sumber daya saat pembuatan. Sumber daya Anda diamankan secara tepat sejak pembuatan—tag yang diterapkan segera ke sumber daya Anda. Oleh karena itu, izin tingkat sumber daya berbasis tag apa pun yang mengontrol penggunaan sumber daya akan langsung diterapkan. Sumber daya Anda dapat dilacak dan dilaporkan dengan lebih akurat. Anda dapat menerapkan penggunaan pemberian tag pada sumber daya baru serta mengontrol kunci dan nilai tag mana yang ditetapkan pada sumber daya Anda.

Untuk informasi selengkapnya, lihat [Contoh pemberian tag sumber daya](#).

Untuk informasi selanjutnya tentang pemberian tag sumber daya Anda untuk penagihan, lihat [Memantau biaya dengan tag alokasi biaya](#).

Menandai cluster dan snapshot

Aturan berikut berlaku untuk pemberian tag sebagai bagian dari operasi permintaan:

- `CreateCluster` :
 - Jika `--cluster-name` disediakan:

Jika tag disertakan dalam permintaan, cluster akan diberi tag.

- Jika `--snapshot-name` disediakan:

Jika tag disertakan dalam permintaan, cluster hanya akan diberi tag dengan tag tersebut. Jika tidak ada tag yang disertakan dalam permintaan, tag snapshot akan ditambahkan ke cluster.

- `CreateSnapshot` :

- Jika `--cluster-name` disediakan:

Jika tag disertakan dalam permintaan, hanya tag permintaan yang akan ditambahkan ke snapshot. Jika tidak ada tag yang disertakan dalam permintaan, tag cluster akan ditambahkan ke snapshot.

- Untuk snapshot otomatis:

Tag akan menyebar dari tag cluster.

- `CopySnapshot` :

Jika tag disertakan dalam permintaan, hanya tag permintaan yang akan ditambahkan ke snapshot. Jika tidak ada tag yang disertakan dalam permintaan, tag snapshot sumber akan ditambahkan ke snapshot salinan.

- `TagResource` dan `UntagResource`:

Tag akan ditambahkan/dihapus dari sumber daya.

Batasan tag

Batasan dasar berikut berlaku untuk tag:

- Jumlah maksimum tag per sumber daya – 50
- Untuk setiap sumber daya, setiap kunci tag harus unik, dan setiap kunci tag hanya dapat memiliki satu nilai.
- Panjang kunci maksimum – 128 karakter Unicode dalam UTF-8.
- Panjang nilai maksimum – 256 karakter Unicode dalam UTF-8.
- Meskipun MemoryDB memungkinkan karakter apa pun dalam tag-nya, layanan lain dapat membatasi. Karakter yang diizinkan di semua layanan adalah huruf, angka, dan spasi yang dapat direpresentasikan dalam UTF-8, serta karakter berikut: `+ - = . _ : / @`
- Kunci dan nilai tag peka huruf besar dan kecil.

- `aws` : Awalan dicadangkan untuk AWS digunakan. Jika tag memiliki kunci tag dengan awalan ini, Anda tidak dapat mengedit atau menghapus kunci atau nilai tag tersebut. Tag dengan awalan `aws` : tidak dihitung terhadap tag per batas sumber daya.

Anda tidak dapat mengakhiri, menghentikan, atau menghapus sumber daya berdasarkan tandanya saja; Anda harus menentukan pengidentifikasi sumber daya tersebut. Misalnya, untuk menghapus snapshot yang Anda beri tag dengan tag kunci yang disebut `DeleteMe`, Anda harus menggunakan tindakan `DeleteSnapshot` dengan pengidentifikasi sumber daya snapshot tersebut, seperti `snap-1234567890abcdef0`.

Untuk informasi lebih lanjut tentang sumber daya MemoryDB yang dapat Anda beri tag, lihat. [Sumber daya yang dapat Anda beri tag](#)

Contoh pemberian tag sumber daya

- Menambahkan tag ke cluster.

```
aws memorydb tag-resource \  
--resource-arn arn:aws:memorydb:us-east-1:111111222233:cluster/my-cluster \  
--tags Key="project",Value="XYZ" Key="memorydb",Value="Service"
```

- Membuat cluster menggunakan tag.

```
aws memorydb create-cluster \  
--cluster-name testing-tags \  
--description cluster-test \  
--subnet-group-name test \  
--node-type db.r6g.large \  
--acl-name open-access \  
--tags Key="project",Value="XYZ" Key="memorydb",Value="Service"
```

- Membuat Snapshot dengan tag.

Untuk kasus ini, jika Anda menambahkan tag berdasarkan permintaan, bahkan jika cluster berisi tag, snapshot hanya akan menerima tag permintaan.

```
aws memorydb create-snapshot \  
--cluster-name testing-tags \  
--snapshot-name bkp-testing-tags-mycluster \  

```

```
--tags Key="work",Value="foo"
```

Memantau biaya dengan tag alokasi biaya

Saat Anda menambahkan tag alokasi biaya ke sumber daya Anda di MemoryDB, Anda dapat melacak biaya dengan mengelompokkan pengeluaran pada faktur Anda berdasarkan nilai tag sumber daya.

Tag alokasi biaya MemoryDB adalah pasangan nilai kunci yang Anda tentukan dan kaitkan dengan sumber daya MemoryDB. Kunci dan nilai peka terhadap huruf besar dan kecil. Anda dapat menggunakan kunci tag untuk menentukan kategori, dan nilai tag dapat berupa item dalam kategori tersebut. Misalnya, Anda dapat menentukan kunci tag `CostCenter` dan nilai tag `10010`, yang menunjukkan bahwa sumber daya tersebut ditetapkan ke pusat pembiayaan 10010. Anda juga dapat menggunakan tag untuk menunjukkan bahwa sumber daya sedang digunakan untuk pengujian atau produksi menggunakan kunci seperti `Environment` dan nilai seperti `test` atau `production`. Sebaiknya gunakan kumpulan kunci tag yang konsisten untuk mempermudah pelacakan biaya yang terkait dengan sumber daya Anda.

Gunakan tag alokasi biaya untuk mengatur AWS tagihan Anda untuk mencerminkan struktur biaya Anda sendiri. Untuk melakukan ini, daftar untuk mendapatkan tagihan AWS akun Anda dengan nilai kunci tag yang disertakan. Kemudian, untuk melihat biaya sumber daya gabungan, atur informasi penagihan Anda sesuai dengan sumber daya Anda dengan nilai kunci tag yang sama. Misalnya, Anda dapat memberikan tag pada beberapa sumber daya dengan nama aplikasi tertentu, kemudian mengatur informasi penagihan untuk melihat biaya total aplikasi tersebut pada beberapa layanan.

Anda juga dapat menggabungkan tag untuk melacak biaya dengan tingkat detail yang lebih besar. Misalnya, untuk melacak biaya layanan Anda menurut wilayah, Anda dapat menggunakan kunci tag `Service` dan `Region`. Di salah satu sumber daya Anda mungkin memiliki nilai `MemoryDB` dan `Asia Pacific (Singapore)`, serta di sumber daya lain Anda mempunyai nilai `MemoryDB` dan `Europe (Frankfurt)`. Anda kemudian dapat melihat total biaya `MemoryDB` Anda dipecah berdasarkan wilayah. Untuk informasi selengkapnya, lihat [Menggunakan Tag Alokasi Biaya](#) dalam Panduan Pengguna AWS Billing .

Anda dapat menambahkan tag alokasi biaya MemoryDB ke cluster MemoryDB. Saat Anda menambahkan, menampilkan daftar, mengubah, menyalin, atau menghapus tag, operasi tersebut hanya akan diterapkan ke klaster yang ditentukan.

Karakteristik tag alokasi biaya MemoryDB

- Tag alokasi biaya diterapkan ke sumber daya MemoryDB yang ditentukan dalam operasi CLI dan API sebagai ARN. Jenis sumber daya akan berupa "klaster".

Format ARN: `arn:aws:memorydb:<region>:<customer-id>:<resource-type>/<resource-name>`

Contoh ARN: `arn:aws:memorydb:us-east-1:1234567890:cluster/my-cluster`

- Kunci tag adalah nama tag yang wajib diisi. Nilai string kunci dapat terdiri dari 1 hingga 128 karakter Unicode dan tidak boleh diawali dengan `aws:`. String dapat berisi hanya kumpulan huruf Unicode, angka, spasi kosong, garis bawah (`_`), titik (`.`), titik dua (`:`), garis miring terbalik (`\`), tanda sama dengan (`=`), tanda plus (`+`), tanda hubung (`-`), atau tanda at (`@`).
- Nilai tag adalah nilai tag opsional. Nilai string dari nilai dapat terdiri dari 1 hingga 256 karakter Unicode dan tidak boleh diawali dengan `aws:`. String dapat berisi hanya kumpulan huruf Unicode, angka, spasi kosong, garis bawah (`_`), titik (`.`), titik dua (`:`), garis miring terbalik (`\`), tanda sama dengan (`=`), tanda plus (`+`), tanda hubung (`-`), atau tanda at (`@`).
- Sumber daya MemoryDB dapat memiliki maksimal 50 tag.
- Nilai tidak harus unik dalam kumpulan tag. Misalnya, Anda dapat memiliki kumpulan tag dengan kunci `Service` dan `Application` yang memiliki nilai `MemoryDB`.

AWS tidak menerapkan makna semantik apa pun pada tag Anda. Tag ditafsirkan secara ketat sebagai string karakter. AWS tidak secara otomatis mengatur tag apa pun pada sumber daya MemoryDB apa pun.

Mengelola tag alokasi biaya Anda menggunakan AWS CLI

Anda dapat menggunakan AWS CLI untuk menambah, memodifikasi, atau menghapus tag alokasi biaya.

Contoh arn: `arn:aws:memorydb:us-east-1:1234567890:cluster/my-cluster`

Topik

- [Listing tag menggunakan AWS CLI](#)
- [Menambahkan tag menggunakan AWS CLI](#)
- [Memodifikasi tag menggunakan AWS CLI](#)

- [Menghapus tag menggunakan AWS CLI](#)

Listing tag menggunakan AWS CLI

Anda dapat menggunakan tag AWS CLI untuk daftar pada sumber daya MemoryDB yang ada dengan menggunakan operasi [list-tag](#).

Kode berikut menggunakan daftar tag pada cluster MemoryDB di wilayah `my-cluster us-east-1`.
AWS CLI

Untuk Linux, macOS, atau Unix:

```
aws memorydb list-tags \  
  --resource-arn arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster
```

Untuk Windows:

```
aws memorydb list-tags ^  
  --resource-arn arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster
```

Output dari operasi ini akan terlihat seperti berikut ini, daftar semua tag pada sumber daya.

```
{  
  "TagList": [  
    {  
      "Value": "10110",  
      "Key": "CostCenter"  
    },  
    {  
      "Value": "EC2",  
      "Key": "Service"  
    }  
  ]  
}
```

Jika tidak ada tag pada sumber daya, output akan kosong TagList.

```
{  
  "TagList": []  
}
```

```
}
```

[Untuk informasi selengkapnya, lihat tag daftar AWS CLI untuk MemoryDB.](#)

Menambahkan tag menggunakan AWS CLI

Anda dapat menggunakan AWS CLI untuk menambahkan tag ke sumber daya MemoryDB yang ada dengan menggunakan operasi CLI [tag-resource](#). Jika kunci tag tidak ada di sumber daya, kunci dan nilai akan ditambahkan ke sumber daya. Jika kunci sudah ada di sumber daya, nilai yang terkait dengan kunci tersebut akan diperbarui ke nilai yang baru.

Kode berikut menggunakan AWS CLI untuk menambahkan kunci Service dan Region dengan nilai-nilai memorydb dan us-east-1 masing-masing ke cluster my-cluster di wilayah us-east-1.

Untuk Linux, macOS, atau Unix:

```
aws memorydb tag-resource \  
  --resource-arn arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster \  
  --tags Key=Service,Value=memorydb \  
         Key=Region,Value=us-east-1
```

Untuk Windows:

```
aws memorydb tag-resource ^  
  --resource-arn arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster ^  
  --tags Key=Service,Value=memorydb ^  
         Key=Region,Value=us-east-1
```

Output dari operasi ini akan terlihat seperti berikut ini, daftar semua tag pada sumber daya mengikuti operasi tersebut.

```
{  
  "TagList": [  
    {  
      "Value": "memorydb",  
      "Key": "Service"  
    },  
    {  
      "Value": "us-east-1",  
      "Key": "Region"  
    }  
  ]  
}
```

```
    }  
  ]  
}
```

Untuk informasi lebih lanjut, lihat AWS CLI untuk [tag-resource](#) MemoryDB.

Anda juga dapat menggunakan AWS CLI untuk menambahkan tag ke cluster saat Anda membuat cluster baru dengan menggunakan operasi [create-cluster](#).

Memodifikasi tag menggunakan AWS CLI

Anda dapat menggunakan AWS CLI untuk memodifikasi tag pada cluster MemoryDB.

Untuk mengubah tag:

- Gunakan [tag-resource](#) untuk menambahkan tag dan nilai baru atau untuk mengubah nilai yang terkait dengan tag yang ada.
- Gunakan [untag-resource](#) untuk menghapus tag tertentu dari sumber daya.

Output dari kedua operasi tersebut akan berupa daftar tag dan nilai-nilainya di klaster yang ditentukan.

Menghapus tag menggunakan AWS CLI

Anda dapat menggunakan AWS CLI untuk menghapus tag dari yang ada dari cluster MemoryDB dengan menggunakan operasi [untag-resource](#).

Kode berikut menggunakan AWS CLI untuk menghapus tag dengan kunci Service dan Region dari cluster `my-cluster` di wilayah `us-east-1`.

Untuk Linux, macOS, atau Unix:

```
aws memorydb untag-resource \  
  --resource-arn arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster \  
  --tag-keys Region Service
```

Untuk Windows:

```
aws memorydb untag-resource ^
```

```
--resource-arn arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster ^  
--tag-keys Region Service
```

Output dari operasi ini akan terlihat seperti berikut ini, daftar semua tag pada sumber daya mengikuti operasi tersebut.

```
{  
  "TagList": []  
}
```

[Untuk informasi selengkapnya, lihat AWS CLI untuk MemoryDB untag-resource.](#)

Mengelola tag alokasi biaya Anda menggunakan API MemoryDB

Anda dapat menggunakan API MemoryDB untuk menambah, memodifikasi, atau menghapus tag alokasi biaya.

Tag alokasi biaya diterapkan ke MemoryDB untuk cluster. Klaster yang akan diberi tag ditentukan menggunakan ARN (Amazon Resource Name).

arn Sampel: `arn:aws:memorydb:us-east-1:1234567890:cluster/my-cluster`

Topik

- [Listing tag menggunakan API MemoryDB](#)
- [Menambahkan tag menggunakan API MemoryDB](#)
- [Memodifikasi tag menggunakan API MemoryDB](#)
- [Menghapus tag menggunakan API MemoryDB](#)

Listing tag menggunakan API MemoryDB

Anda dapat menggunakan API MemoryDB untuk mencantumkan tag pada sumber daya yang ada dengan menggunakan operasi. [ListTags](#)

Kode berikut menggunakan API MemoryDB untuk mencantumkan tag pada sumber daya `my-cluster` di wilayah `us-east-1`.

```
https://memory-db.us-east-1.amazonaws.com/  
?Action=ListTags
```

```
&ResourceArn=arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Version=2021-01-01
&Timestamp=20210802T192317Z
&X-Amz-Credential=<credential>
```

Menambahkan tag menggunakan API MemoryDB

Anda dapat menggunakan API MemoryDB untuk menambahkan tag ke cluster MemoryDB yang ada dengan menggunakan operasi [TagResource](#). Jika kunci tag tidak ada di sumber daya, kunci dan nilai akan ditambahkan ke sumber daya. Jika kunci sudah ada di sumber daya, nilai yang terkait dengan kunci tersebut akan diperbarui ke nilai yang baru.

Kode berikut menggunakan API MemoryDB untuk menambahkan kunci Service dan Region dengan nilai memorydb dan us-east-1 masing-masing ke sumber daya my-cluster di wilayah us-east-1.

```
https://memory-db.us-east-1.amazonaws.com/
?Action=TagResource
&ResourceArn=arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Tags.member.1.Key=Service
&Tags.member.1.Value=memorydb
&Tags.member.2.Key=Region
&Tags.member.2.Value=us-east-1
&Version=2021-01-01
&Timestamp=20210802T192317Z
&X-Amz-Credential=<credential>
```

Untuk informasi lebih lanjut, lihat [TagResource](#).

Memodifikasi tag menggunakan API MemoryDB

Anda dapat menggunakan API MemoryDB untuk memodifikasi tag pada cluster MemoryDB.

Untuk mengubah nilai tag:

- Gunakan [TagResource](#) operasi untuk menambahkan tag dan nilai baru atau untuk mengubah nilai tag yang ada.

- Gunakan [UntagResource](#) untuk menghapus tag dari sumber daya.

Hasil dari kedua operasi tersebut akan berupa daftar tag dan nilai-nilainya di sumber daya yang ditentukan.

Menghapus tag menggunakan API MemoryDB

Anda dapat menggunakan API MemoryDB untuk menghapus tag dari cluster MemoryDB yang ada dengan menggunakan operasi. [UntagResource](#)

Kode berikut menggunakan API MemoryDB untuk menghapus tag dengan kunci Service dan Region dari cluster `my-cluster` di wilayah `us-east-1`.

```
https://memory-db.us-east-1.amazonaws.com/  
?Action=UntagResource  
&ResourceArn=arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&TagKeys.member.1=Service  
&TagKeys.member.2=Region  
&Version=2021-01-01  
&Timestamp=20210802T192317Z  
&X-Amz-Credential=<credential>
```

Mengelola pemeliharaan

Setiap klaster memiliki periode pemeliharaan mingguan saat perubahan sistem akan diterapkan. Jika Anda tidak menentukan jendela pemeliharaan yang disukai saat membuat atau memodifikasi klaster, MemoryDB menetapkan jendela pemeliharaan 60 menit dalam jendela pemeliharaan wilayah Anda pada hari yang dipilih secara acak dalam seminggu.

Periode pemeliharaan 60 menit dipilih secara acak dari blok waktu 8 jam per wilayah. Tabel berikut menampilkan daftar blok waktu untuk setiap wilayah yang akan digunakan untuk menetapkan periode pemeliharaan default. Anda dapat memilih periode pemeliharaan yang diinginkan di luar blok periode pemeliharaan wilayah.

Kode Wilayah	Nama wilayah	Periode Pemeliharaan Wilayah
ap-northeast-1	Wilayah Asia Pasifik (Tokyo)	13.00–21.00 UTC

Kode Wilayah	Nama wilayah	Periode Pemeliharaan Wilayah
ap-northeast-2	Wilayah Asia Pasifik (Seoul)	12.00–20.00 UTC
ap-south-1	Wilayah Asia Pasifik (Mumbai)	17.30–01.30 UTC
ap-southeast-1	Wilayah Asia Pasifik (Singapura)	14.00–22.00 UTC
ap-east-1	Wilayah Asia Pasifik (Hong Kong)	13.00–21.00 UTC
ap-southeast-2	Wilayah Asia Pasifik (Sydney)	12.00–20.00 UTC
cn-north-1	Wilayah Tiongkok (Beijing)	14.00–22.00 UTC
cn-northwest-1	Wilayah Tiongkok (Ningxia)	14.00–22.00 UTC
eu-west-3	Wilayah Eropa (Paris)	23.59–07.29 UTC
eu-central-1	Wilayah Eropa (Frankfurt)	23.00–07.00 UTC
eu-west-1	Wilayah Eropa (Irlandia)	22.00–06.00 UTC
eu-west-2	Wilayah Eropa (London)	23.00–07.00 UTC
sa-east-1	Wilayah South America (São Paulo)	01.00–09.00 UTC
ca-central-1	Wilayah Kanada (Pusat)	03.00–11.00 UTC
us-east-1	Wilayah AS Timur (N. Virginia)	03.00–11.00 UTC
us-east-1	Wilayah AS Timur (Ohio)	04.00–12.00 UTC
us-west-1	Wilayah US West (N. California)	06.00–14.00 UTC
us-west-2	Wilayah US West (Oregon)	06.00–14.00 UTC

Mengubah periode pemeliharaan Klaster Anda

Periode pemeliharaan harus berada dalam waktu penggunaan terendah, sehingga kemungkinan memerlukan perubahan dari waktu ke waktu. Anda dapat mengubah klaster Anda untuk menentukan

rentang waktu hingga durasi 24 jam saat aktivitas pemeliharaan yang Anda minta akan dilakukan. Setiap perubahan klaster yang Anda minta yang ditangguhkan atau ditunda akan terjadi selama waktu ini.

Informasi lain

Untuk informasi tentang periode pemeliharaan dan penggantian simpul, lihat yang berikut ini:

- [Mengganti simpul](#)—Mengelola penggantian simpul
- [Memodifikasi cluster MemoryDB](#)—Mengubah periode pemeliharaan klaster

Praktik terbaik

Berikut ini, Anda dapat menemukan praktik terbaik yang direkomendasikan untuk MemoryDB. Mengikuti langkah ini meningkatkan kinerja dan keandalan klaster Anda.

Topik

- [Perintah Redis OSS Terbatas](#)
- [Ketahanan dalam MemoryDB](#)
- [Praktik terbaik: Pub/Sub dan Multiplexing I/O yang Ditingkatkan](#)
- [Praktik terbaik: Mengubah ukuran klaster secara online](#)

Perintah Redis OSS Terbatas

Untuk memberikan pengalaman layanan terkelola, MemoryDB membatasi akses ke perintah tertentu yang memerlukan hak istimewa tingkat lanjut. Perintah berikut tidak tersedia:

- `acl deluser`
- `acl load`
- `acl save`
- `acl setuser`
- `bgrewriteaof`
- `bgsave`
- `cluster addslot`
- `cluster delslot`
- `cluster setslot`
- `config`
- `debug`
- `migrate`
- `module`
- `psync`
- `replicaof`
- `save`
- `shutdown`
- `slaveof`
- `sync`

Ketahanan dalam MemoryDB

Infrastruktur AWS global dibangun di sekitar AWS Wilayah dan Zona Ketersediaan. AWS Wilayah menyediakan beberapa Availability Zone yang terpisah secara fisik dan terisolasi, yang terhubung dengan latensi rendah, throughput tinggi, dan jaringan yang sangat redundan. Dengan Zona Ketersediaan, Anda dapat merancang serta mengoperasikan aplikasi dan basis data yang secara otomatis melakukan failover di antara Zona Ketersediaan tanpa gangguan. Zona Ketersediaan memiliki ketersediaan dan toleransi kesalahan yang lebih baik, dan dapat diskalakan dibandingkan infrastruktur biasa yang terdiri dari satu atau beberapa pusat data.

Untuk informasi selengkapnya tentang AWS Wilayah dan Availability Zone, lihat [Infrastruktur AWS Global](#).

Selain infrastruktur AWS global, MemoryDB menawarkan beberapa fitur untuk membantu mendukung ketahanan data dan kebutuhan snapshot Anda.

Topik

- [Mitigasi Kegagalan](#)

Mitigasi Kegagalan

Saat merencanakan implementasi MemoryDB Anda, Anda harus merencanakan sehingga kegagalan memiliki dampak minimal pada aplikasi dan data Anda. Topik pada bagian ini membahas pendekatan yang dapat Anda ambil untuk melindungi aplikasi dan data Anda dari kegagalan.

Mengurangi Kegagalan: Cluster MemoryDB

Cluster MemoryDB terdiri dari satu node utama yang aplikasi Anda dapat membaca dari dan menulis ke, dan dari 0 hingga 5 node replika read-only. Namun, kami sangat menyarankan untuk menggunakan setidaknya 1 replika untuk ketersediaan tinggi. Setiap kali data ditulis ke node utama, data tersebut disimpan ke log transaksi dan diperbarui secara asinkron pada node replika.

Saat salah satu replika baca gagal

1. MemoryDB mendeteksi replika yang gagal.
2. MemoryDB membuat node yang gagal offline.
3. MemoryDB meluncurkan dan menyediakan node pengganti di AZ yang sama.
4. Node baru disinkronkan dengan log transaksi.

Selama proses ini, aplikasi Anda dapat terus melakukan pembacaan dan penulisan menggunakan simpul lain.

MemoryDB Multi-AZ

Jika Multi-AZ diaktifkan pada cluster MemoryDB Anda, primer yang gagal akan terdeteksi dan diganti secara otomatis.

1. MemoryDB mendeteksi kegagalan node utama.
2. MemoryDB gagal ke replika setelah memastikannya konsisten dengan primer yang gagal.
3. MemoryDB memutar replika di AZ primer yang gagal.
4. Node baru disinkronkan dengan log transaksi.

Melakukan failover ke simpul replika umumnya lebih cepat daripada membuat dan menyediakan simpul primer baru. Ini berarti aplikasi Anda dapat melanjutkan penulisan ke node utama Anda lebih cepat.

Untuk informasi selengkapnya, lihat [Meminimalkan downtime di MemoryDB dengan Multi-AZ](#).

Praktik terbaik: Pub/Sub dan Multiplexing I/O yang Ditingkatkan

Saat menggunakan Redis OSS versi 7 atau yang lebih baru, kami sarankan menggunakan [sharded Pub/Sub](#). Anda juga meningkatkan throughput dan latensi menggunakan [multiplexing I/O yang disempurnakan](#), yang secara otomatis tersedia saat menggunakan Redis OSS versi 7 atau yang lebih baru dan tidak memerlukan perubahan klien. Ini sangat ideal untuk beban kerja pub/sub, yang seringkali terikat throughput dengan beberapa koneksi klien.

Praktik terbaik: Mengubah ukuran klaster secara online

Resharding melibatkan proses menambahkan dan menghapus serpihan atau simpul ke klaster Anda dan mendistribusikan ulang keyspace. Oleh karena itu, ada beberapa hal yang memiliki dampak pada operasi resharding, seperti beban pada klaster, pemanfaatan memori, dan ukuran keseluruhan data. Untuk pengalaman terbaik, sebaiknya ikuti praktik terbaik klaster secara keseluruhan untuk distribusi pola beban kerja seragam. Selain itu, sebaiknya lakukan langkah-langkah berikut.

Sebelum memulai resharding, sebaiknya lakukan hal berikut:

- Uji aplikasi Anda – Uji perilaku aplikasi Anda selama resharding di lingkungan staging jika memungkinkan.
- Dapatkan notifikasi awal untuk masalah penskalaan – Resharding adalah operasi sarat komputasi. Karena itu, kami merekomendasikan untuk menjaga pemanfaatan CPU di bawah 80 persen pada instance multicore dan kurang dari 50 persen pada instance inti tunggal selama resharding. Pantau metrik MemoryDB dan memulai resharding sebelum aplikasi Anda mulai mengamati masalah penskalaan. Metrik yang berguna untuk dilacak adalah CPUUtilization, NetworkBytesIn, NetworkBytesOut, CurrConnections, NewConnections, FreeableMemory, SwapUsage, dan BytesUsedForMemoryDB.
- Pastikan untuk menyediakan memori yang cukup sebelum penskalaan ke dalam – Jika Anda melakukan penskalaan ke dalam, pastikan memori pada serpihan yang akan dipertahankan tersedia setidaknya 1,5 kali dari memori yang akan digunakan pada serpihan yang akan dihapus.
- Mulai resharding di luar jam puncak – Praktik ini membantu mengurangi dampak latensi dan throughput pada klien selama operasi resharding. Hal ini juga membantu untuk menyelesaikan resharding lebih cepat karena lebih banyak sumber daya dapat digunakan untuk distribusi ulang slot.
- Tinjau perilaku waktu habis klien – Beberapa klien mungkin mengalami latensi yang lebih tinggi selama perubahan ukuran klaster online. Anda dapat mengonfigurasi pustaka klien dengan waktu habis yang lebih tinggi agar sistem memiliki waktu untuk terhubung bahkan dalam kondisi beban

yang lebih tinggi pada server. Dalam beberapa kasus, mungkin sebaiknya buka sejumlah besar koneksi ke server. Dalam kasus ini, pertimbangkan untuk menambahkan backoff eksponensial ke logika koneksi ulang. Melakukan hal ini dapat membantu mencegah lonjakan koneksi baru di server pada waktu yang sama.

Selama resharding, sebaiknya melakukan hal berikut:

- Hindari perintah yang menghabiskan banyak daya komputasi – Hindari menjalankan operasi yang sarat komputasi dan I/O, seperti perintah KEYS dan SMEMBERS. Pendekatan ini disarankan karena operasi ini meningkatkan beban pada klaster dan memiliki dampak pada performa klaster. Sebagai gantinya, gunakan perintah SCAN dan SSCAN.
- Ikuti praktik terbaik Lua – Hindari menjalankan skrip Lua terlalu lama, dan selalu nyatakan kunci yang digunakan dalam skrip Lua di depan. Pendekatan ini disarankan untuk menentukan bahwa skrip Lua tidak menggunakan perintah cross slot. Pastikan bahwa kunci yang digunakan dalam skrip Lua adalah milik slot yang sama.

Setelah resharding, perhatikan hal berikut:

- Penskalaan ke dalam mungkin berhasil sebagian jika memori tidak cukup pada serpihan target. Jika hasil seperti itu terjadi, tinjau memori yang tersedia dan coba lagi operasi itu, jika perlu.
- Slot dengan item besar tidak dimigrasikan. Secara khusus, slot dengan item yang lebih besar dari 256 MB pasca-serialisasi tidak dimigrasikan.
- Perintah FLUSHALL dan FLUSHDB tidak didukung dalam skrip Lua selama operasi resharding.

Memahami replikasi MemoryDB

MemoryDB mengimplementasikan replikasi dengan data yang dipartisi hingga 500 pecahan.

Setiap serpihan di dalam klaster memiliki simpul primer baca/tulis tunggal dan hingga 5 simpul replika baca-saja. Setiap node utama dapat mempertahankan hingga 100 MB/s. Anda dapat membuat klaster dengan jumlah serpihan lebih banyak dan jumlah replika lebih sedikit dengan jumlah hingga 500 simpul per klaster. Konfigurasi klaster ini dapat berkisar dari 500 serpihan dan 0 replika hingga 100 serpihan dan 4 replika, yang merupakan jumlah replika maksimum yang diperbolehkan.

Konsistensi

Dalam MemoryDB, node primer sangat konsisten. Operasi penulisan yang berhasil disimpan secara tahan lama dalam log transaksional Multi-AZ terdistribusi sebelum kembali ke klien. Operasi baca pada pendahuluan selalu mengembalikan sebagian besar up-to-date data yang mencerminkan efek dari semua operasi penulisan yang berhasil sebelumnya. Konsistensi yang kuat seperti itu dipertahankan di seluruh failovers primer.

Di MemoryDB, simpul replika pada akhirnya konsisten. Operasi baca dari replika (menggunakan READONLY perintah) mungkin tidak selalu mencerminkan efek operasi penulisan terbaru yang berhasil, dengan metrik lag yang dipublikasikan. CloudWatch Namun, operasi baca dari satu replika konsisten secara berurutan. Operasi menulis sukses berlaku pada setiap replika dalam urutan yang sama mereka dieksekusi pada primer.

Replikasi dalam sebuah cluster

Setiap replika baca dalam serpihan memelihara salinan data dari simpul primer pada serpihan. Mekanisme replikasi asinkron menggunakan log transaksi digunakan untuk menjaga sinkronisasi replika baca dengan primer. Aplikasi dapat membaca dari simpul apa pun di dalam kluster. Aplikasi hanya dapat menulis ke simpul primer. Replika baca meningkatkan skalabilitas baca. Sejak MemoryDB menyimpan data dalam log transaksi tahan lama, tidak ada risiko bahwa data akan hilang. Data dipartisi di seluruh serpihan di dalam kluster MemoryDB.

Aplikasi menggunakan titik akhir kluster MemoryDB kluster untuk terhubung dengan simpul di dalam kluster. Untuk informasi selengkapnya, lihat [Menemukan titik akhir koneksi](#).

cluster MemoryDB bersifat regional dan dapat berisi node hanya dari satu Region. Untuk meningkatkan toleransi kesalahan, Anda harus menyediakan primer dan membaca replika di beberapa Availability Zones di beberapa Availability Zones di dalam wilayah tersebut.

Menggunakan replikasi, yang menyediakan Anda dengan Multi-AZ, sangat dianjurkan untuk semua cluster MemoryDB. Untuk informasi selengkapnya, lihat [Meminimalkan downtime di MemoryDB dengan Multi-AZ](#).

Meminimalkan downtime di MemoryDB dengan Multi-AZ

Ada sejumlah contoh di mana MemoryDB mungkin perlu mengganti node primer; ini termasuk jenis pemeliharaan terencana tertentu dan kejadian yang tidak mungkin dari node primer atau kegagalan Availability Zone.

Respon terhadap kegagalan node tergantung pada node mana yang gagal. Namun, dalam semua kasus, MemoryDB memastikan bahwa tidak ada data yang hilang selama penggantian node atau failover. Misalnya, jika replika gagal, node yang gagal diganti dan data disinkronkan dari log transaksi. Jika node utama gagal, failover dipicu ke replika konsisten yang memastikan tidak ada data yang hilang selama failover. Penulisan sekarang dilayani dari simpul utama baru. Node primer lama kemudian diganti dan disinkronkan dari log transaksi.

Jika node primer gagal pada pecahan node tunggal (tidak ada replika), MemoryDB berhenti menerima penulisan sampai node utama diganti dan disinkronkan dari log transaksi.

Penggantian node dapat mengakibatkan beberapa downtime untuk cluster, tetapi jika Multi-AZ aktif, downtime diminimalkan. Peran node primer akan secara otomatis gagal ke salah satu replika. Tidak perlu membuat dan menyediakan simpul utama baru, karena MemoryDB akan menangani ini secara transparan. Failover dan promosi replika ini memastikan Anda dapat melanjutkan penulisan ke primer baru segera setelah promosi selesai.

Dalam kasus penggantian node yang direncanakan dimulai karena pembaruan pemeliharaan atau pembaruan layanan, perhatikan penggantian node yang direncanakan selesai saat cluster melayani permintaan tulis yang masuk.

Multi-AZ pada cluster MemoryDB Anda meningkatkan toleransi kesalahan Anda. Ini benar terutama dalam kasus di mana node utama cluster Anda menjadi tidak dapat dijangkau atau gagal karena alasan apa pun. Multi-AZ pada cluster MemoryDB mengharuskan setiap pecahan memiliki lebih dari satu node, dan diaktifkan secara otomatis.

Topik

- [Skenario kegagalan dengan respons Multi-AZ](#)
- [Menguji failover otomatis](#)

Skenario kegagalan dengan respons Multi-AZ

Jika Multi-AZ aktif, node primer yang gagal gagal ke replika yang tersedia. Replika secara otomatis disinkronkan dengan log transaksi dan menjadi primer, yang jauh lebih cepat daripada membuat dan

reprovisioning node primer baru. Proses ini biasanya memakan waktu hanya beberapa detik hingga Anda dapat menulis lagi ke klaster.

Ketika Multi-AZ aktif, MemoryDB terus memantau keadaan node utama. Jika simpul primer gagal, salah satu tindakan berikut akan dilakukan bergantung pada jenis kegagalan.

Topik

- [Skenario kegagalan ketika hanya simpul primer yang gagal](#)
- [Skenario kegagalan ketika node utama dan beberapa replika gagal](#)
- [Skenario kegagalan ketika seluruh klaster gagal](#)

Skenario kegagalan ketika hanya simpul primer yang gagal

Jika hanya node primer yang gagal, replika akan secara otomatis menjadi primer. Replika pengganti kemudian dibuat dan disediakan di Availability Zone yang sama dengan primer yang gagal.

Ketika hanya node utama yang gagal, MemoryDB Multi-AZ melakukan hal berikut:

1. Simpul primer yang gagal akan dibuat offline.
2. up-to-date Replika secara otomatis menjadi primer.

Menulis dapat dilanjutkan segera setelah proses failover selesai, biasanya hanya beberapa detik.

3. Replika pengganti diluncurkan dan disediakan.

Replika pengganti diluncurkan di Availability Zone tempat node utama yang gagal berada sehingga distribusi node dipertahankan.

4. Replika disinkronkan dengan log transaksi.

Untuk informasi tentang cara menemukan titik akhir klaster, lihat topik berikut:

- [Menemukan Endpoint untuk Cluster MemoryDB \(MemoryDB API\)](#)

Skenario kegagalan ketika node utama dan beberapa replika gagal

Jika replika primer dan setidaknya satu replika gagal, up-to-date replika dipromosikan ke cluster primer. Replika baru juga dibuat dan disediakan di Availability Zone yang sama dengan node yang gagal.

Ketika node utama dan beberapa replika gagal, MemoryDB Multi-AZ melakukan hal berikut:

1. Node primer yang gagal dan replika yang gagal diambil offline.
2. Replika yang tersedia akan menjadi simpul utama.

Menulis dapat dilanjutkan segera setelah failover selesai, biasanya hanya beberapa detik.

3. Replika pengganti dibuat dan ditetapkan.

Replika pengganti dibuat di Zona Ketersediaan dari simpul yang gagal sehingga distribusi simpul tetap terpelihara.

4. Semua node disinkronkan dengan log transaksi.

Untuk informasi tentang cara menemukan titik akhir klaster, lihat topik berikut:

- [Menemukan Titik Akhir untuk Cluster MemoryDB \(CLI\)AWS](#)
- [Menemukan Endpoint untuk Cluster MemoryDB \(MemoryDB API\)](#)

Skenario kegagalan ketika seluruh klaster gagal

Jika semuanya gagal, semua simpul dibuat kembali dan ditetapkan pada Zona Ketersediaan yang sama dengan simpul asli.

Tidak ada kehilangan data dalam skenario ini karena data disimpan dalam log transaksi.

Ketika seluruh cluster gagal, MemoryDB Multi-AZ melakukan hal berikut:

1. Node primer dan replika yang gagal diambil offline.
2. Node primer pengganti dibuat dan disediakan, disinkronkan dengan log transaksi.
3. Replika pengganti dibuat dan disediakan, disinkronkan dengan log transaksi.

Penggantinya dibuat di Zona Ketersediaan dari simpul yang gagal sehingga distribusi simpul tetap dipertahankan.

Untuk informasi tentang cara menemukan titik akhir klaster, lihat topik berikut:

- [Menemukan Titik Akhir untuk Cluster MemoryDB \(CLI\)AWS](#)
- [Menemukan Endpoint untuk Cluster MemoryDB \(MemoryDB API\)](#)

Menguji failover otomatis

Anda dapat menguji failover otomatis menggunakan konsol MemoryDB, API AWS CLI, dan MemoryDB.

Saat menguji, perhatikan hal berikut:

- Anda dapat menggunakan operasi ini hingga lima kali dalam periode 24 jam apa pun.
- Jika Anda memanggil operasi ini pada pecahan di cluster yang berbeda, Anda dapat melakukan panggilan secara bersamaan.
- Dalam beberapa kasus, Anda mungkin memanggil operasi ini beberapa kali pada pecahan yang berbeda di cluster MemoryDB yang sama. Dalam kasus tersebut, penggantian simpul pertama harus selesai sebelum panggilan berikutnya dapat dibuat.
- Untuk menentukan apakah penggantian node selesai, periksa peristiwa menggunakan konsol MemoryDB, API AWS CLI, atau MemoryDB. Cari peristiwa berikut yang terkait dengan `FailoverShard`, tercantum di sini dalam urutan kemungkinan terjadinya:
 1. pesan cluster: `FailoverShard API called for shard <shard-id>`
 2. pesan cluster: `Failover from primary node <primary-node-id> to replica node <node-id> completed`
 3. pesan cluster: `Recovering nodes <node-id>`
 4. pesan cluster: `Finished recovery for nodes <node-id>`

Untuk informasi selengkapnya, lihat berikut ini:

- [DescribeEvents](#) di Referensi API MemoryDB
- API ini dirancang untuk menguji perilaku aplikasi Anda jika terjadi failover MemoryDB. Hal ini tidak dirancang untuk menjadi alat operasional untuk memulai failover guna mengatasi masalah dengan klaster. Selain itu, dalam kondisi tertentu seperti peristiwa operasional skala besar, AWS dapat memblokir API ini.

Topik

- [Menguji failover otomatis menggunakan AWS Management Console](#)
- [Menguji failover otomatis menggunakan AWS CLI](#)
- [Menguji failover otomatis menggunakan API MemoryDB](#)

Menguji failover otomatis menggunakan AWS Management Console

Gunakan prosedur berikut untuk menguji failover otomatis dengan konsol.

1. [Masuk ke AWS Management Console dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/](https://console.aws.amazon.com/memorydb/).
2. Pilih tombol radio di sebelah kiri cluster yang ingin Anda uji. Cluster ini harus memiliki setidaknya satu node replika.
3. Pada bagian Detail, lakukan konfirmasi bahwa klaster ini sudah mengaktifkan Multi-AZ. Jika klaster tidak memiliki Multi-AZ yang aktif, pilih klaster yang berbeda atau ubah klaster ini agar memiliki Multi-AZ yang aktif. Untuk informasi selengkapnya, lihat [Memodifikasi cluster MemoryDB](#).
4. Pilih nama klaster.
5. Pada halaman Pecahan dan node, untuk pecahan yang ingin Anda uji failover, pilih nama pecahan.
6. Untuk node, pilih Failover Primary.
7. Pilih Lanjutkan untuk melakukan failover primer, atau Batalkan untuk membatalkan failover simpul primer.

Selama proses failover, konsol terus menunjukkan status simpul sebagai tersedia. Untuk memantau progres pengujian failover Anda, pilih Peristiwa dari panel navigasi konsol. Di tab Peristiwa, perhatikan peristiwa yang menunjukkan failover Anda telah dimulai (FailoverShard API called) dan selesai (Recovery completed).

Menguji failover otomatis menggunakan AWS CLI

[Anda dapat menguji failover otomatis pada klaster berkemampuan multi-AZ apa pun menggunakan AWS CLI operasi failover-shard.](#)

Parameter

- `--cluster-name` – Wajib. Cluster yang akan diuji.
- `--shard-name` – Wajib. Nama pecahan yang ingin Anda uji failover otomatis. Anda dapat menguji maksimal lima pecahan dalam periode 24 jam bergulir.

Contoh berikut menggunakan AWS CLI to call `failover-shard` on the shard `0001` di cluster MemoryDB. `my-cluster`

Untuk Linux, macOS, atau Unix:

```
aws memorydb failover-shard \  
  --cluster-name my-cluster \  
  --shard-name 0001
```

Untuk Windows:

```
aws memorydb failover-shard ^  
  --cluster-name my-cluster ^  
  --shard-name 0001
```

Untuk melacak kemajuan failover Anda, gunakan AWS CLI `describe-events` operasi.

Ini akan mengembalikan respons JSON berikut:

```
{  
  "Events": [  
    {  
      "SourceName": "my-cluster",  
      "SourceType": "cluster",  
      "Message": "Failover to replica node my-cluster-0001-002 completed",  
      "Date": "2021-08-22T12:39:37.568000-07:00"  
    },  
    {  
      "SourceName": "my-cluster",  
      "SourceType": "cluster",  
      "Message": "Starting failover for shard 0001",  
      "Date": "2021-08-22T12:39:10.173000-07:00"  
    }  
  ]  
}
```

Untuk informasi selengkapnya, lihat berikut ini:

- [failover-shard](#)
- [describe-events](#)

Menguji failover otomatis menggunakan API MemoryDB

Contoh berikut memanggil pecahan `FailoverShard 0003` di `clustermemorydb00`.

Example Menguji failover otomatis

```
https://memory-db.us-east-1.amazonaws.com/  
  ?Action=FailoverShard  
  &ShardName=0003  
  &ClusterName=memorydb00  
  &Version=2021-01-01  
  &SignatureVersion=4  
  &SignatureMethod=HmacSHA256  
  &Timestamp=20210801T192317Z  
  &X-Amz-Credential=<credential>
```

Untuk melacak kemajuan failover Anda, gunakan operasi API `MemoryDBDescribeEvents`.

Untuk informasi selengkapnya, lihat informasi berikut:

- [FailoverShard](#)
- [DescribeEvents](#)

Mengubah jumlah replika

Anda dapat secara dinamis menambah atau mengurangi jumlah replika baca dalam kluster MemoryDB Anda menggunakan AWS Management Console, atau API MemoryDB. AWS CLI Semua serpihan harus memiliki jumlah replika yang sama.

Menambah jumlah replika dalam klaster

Anda dapat menambah jumlah replika di dalam klaster MemoryDB maksimum hingga lima per serpihan. Anda dapat melakukannya menggunakan AWS Management Console, AWS CLI, atau API MemoryDB.

Topik

- [Menggunakan AWS Management Console](#)
- [Menggunakan AWS CLI](#)
- [Menggunakan API MemoryDB](#)

Menggunakan AWS Management Console

Untuk menambah jumlah replika di dalam klaster (console) MemoryDB, lihat. [Menambahkan/Menghapus node dari cluster](#)

Menggunakan AWS CLI

Untuk menambah jumlah replika di dalam klaster MemoryDB, gunakan `update-cluster` perintah dengan parameter berikut:

- `--cluster-name` – Diperlukan. Mengidentifikasi klaster yang ingin ditambah jumlah replika di dalamnya.
- `--replica-configuration` – Diperlukan. Memungkinkan Anda mengatur jumlah replika. Untuk menambah jumlah replika, atur `ReplicaCount` properti ke jumlah replika yang Anda inginkan di dalam serpihan ini pada akhir operasi ini.

Example

Contoh berikut menambah jumlah replika di dalam klaster `my-cluster` menjadi 2.

Untuk Linux, macOS, atau Unix:

```
aws memorydb update-cluster \  
  --cluster-name my-cluster \  
  --replica-configuration \  
    ReplicaCount=2
```

Untuk Windows:

```
aws memorydb update-cluster ^
  --cluster-name my-cluster ^
  --replica-configuration ^
    ReplicaCount=2
```

la mengembalikan respon JSON berikut:

```
{
  "Cluster": {
    "Name": "my-cluster",
    "Status": "updating",
    "NumberOfShards": 1,
    "ClusterEndpoint": {
      "Address": "clustercfg.my-cluster.xxxxx.memorydb.us-east-1.amazonaws.com",
      "Port": 6379
    },
    "NodeType": "db.r6g.large",
    "EngineVersion": "6.2",
    "EnginePatchVersion": "6.2.6",
    "ParameterGroupName": "default.memorydb-redis6",
    "ParameterGroupStatus": "in-sync",
    "SubnetGroupName": "my-sg",
    "TLSEnabled": true,
    "ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",
    "SnapshotRetentionLimit": 0,
    "MaintenanceWindow": "wed:03:00-wed:04:00",
    "SnapshotWindow": "04:30-05:30",
    "DataTiering": "false",
    "AutoMinorVersionUpgrade": true
  }
}
```

Untuk melihat rincian klaster yang diperbarui setelah statusnya berubah dari memperbarui ke tersedia, gunakan perintah berikut:

Untuk Linux, macOS, atau Unix:

```
aws memorydb describe-clusters \
  --cluster-name my-cluster
  --show-shard-details
```

Untuk Windows:

```
aws memorydb describe-clusters ^
  --cluster-name my-cluster
  --show-shard-details
```

Ini akan mengembalikan respon JSON berikut:

```
{
  "Clusters": [
    {
      "Name": "my-cluster",
      "Status": "available",
      "NumberOfShards": 1,
      "Shards": [
        {
          "Name": "0001",
          "Status": "available",
          "Slots": "0-16383",
          "Nodes": [
            {
              "Name": "my-cluster-0001-001",
              "Status": "available",
              "AvailabilityZone": "us-east-1a",
              "CreateTime": "2021-08-21T20:22:12.405000-07:00",
              "Endpoint": {
                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                "Port": 6379
              }
            },
            {
              "Name": "my-cluster-0001-002",
              "Status": "available",
              "AvailabilityZone": "us-east-1b",
              "CreateTime": "2021-08-21T20:22:12.405000-07:00",
              "Endpoint": {
                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                "Port": 6379
              }
            },
            {
              "Name": "my-cluster-0001-003",
```

```

        "Status": "available",
        "AvailabilityZone": "us-east-1a",
        "CreateTime": "2021-08-22T12:59:31.844000-07:00",
        "Endpoint": {
            "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
            "Port": 6379
        }
    ],
    "NumberOfNodes": 3
}
],
"ClusterEndpoint": {
    "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
    "Port": 6379
},
"NodeType": "db.r6g.large",
"EngineVersion": "6.2",
"EnginePatchVersion": "6.2.6",
"ParameterGroupName": "default.memorydb-redis6",
"ParameterGroupStatus": "in-sync",
"SubnetGroupName": "my-sg",
"TLSEnabled": true,
"ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",
"SnapshotRetentionLimit": 0,
"MaintenanceWindow": "wed:03:00-wed:04:00",
"SnapshotWindow": "04:30-05:30",
"ACLName": "my-acl",
"DataTiering": "false",
"AutoMinorVersionUpgrade": true
}
]
}

```

Untuk informasi lebih lanjut tentang meningkatkan jumlah replika menggunakan CLI, lihat [update-cluster](#) di Command Reference. AWS CLI

Menggunakan API MemoryDB

Untuk menambah jumlah replika di dalam serpihan MemoryDB, gunakan `UpdateCluster` tindakan dengan parameter berikut:

- `ClusterName` – Diperlukan. Mengidentifikasi klaster yang ingin ditambah jumlah replika di dalamnya.
- `ReplicaConfiguration` – Diperlukan. Memungkinkan Anda mengatur jumlah replika. Untuk menambah jumlah replika, atur `ReplicaCount` properti ke jumlah replika yang Anda inginkan di dalam serpihan ini pada akhir operasi ini.

Example

Contoh berikut menambah jumlah replika di dalam klaster `sample-cluster` menjadi tiga. Saat contoh ini selesai, terdapat tiga replika di setiap serpihan. Nomor ini berlaku apakah ini adalah cluster MemoryDB dengan pecahan tunggal atau cluster MemoryDB dengan beberapa pecahan.

```
https://memory-db.us-east-1.amazonaws.com/  
?Action=UpdateCluster  
&ReplicaConfiguration.ReplicaCount=3  
&ClusterName=sample-cluster  
&Version=2021-01-01  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20210802T192317Z  
&X-Amz-Credential=<credential>
```

Untuk informasi lain tentang menambah jumlah replika menggunakan API, lihat [UpdateCluster](#).

Mengurangi jumlah replika dalam klaster

Anda dapat mengurangi jumlah replika di dalam klaster untuk MemoryDB. Anda dapat mengurangi jumlah replika menjadi nol, tetapi Anda tidak dapat failover ke replika jika simpul primer Anda gagal.

Anda dapat menggunakan AWS Management Console, AWS CLI atau API MemoryDB untuk mengurangi jumlah replika di dalam klaster.

Topik

- [Menggunakan AWS Management Console](#)
- [Menggunakan AWS CLI](#)
- [Menggunakan API MemoryDB](#)

Menggunakan AWS Management Console

Untuk mengurangi jumlah replika di dalam klaster (console) MemoryDB, lihat. [Menambahkan/Menghapus node dari cluster](#)

Menggunakan AWS CLI

Untuk mengurangi jumlah replika di dalam klaster MemoryDB, gunakan `update-cluster` perintah dengan parameter berikut:

- `--cluster-name` – Diperlukan. Mengidentifikasi klaster yang ingin dikurangi jumlah replika di dalamnya.
- `--replica-configuration` – Diperlukan.

`ReplicaCount`- Mengatur properti ini untuk menentukan jumlah simpul replika yang Anda inginkan.

Example

Contoh berikut menggunakan `--replica-configuration` untuk mengurangi jumlah replika di dalam klaster dengan nilai `my-cluster` yang ditentukan.

Untuk Linux, macOS, atau Unix:

```
aws memorydb update-cluster \
```

```
--cluster-name my-cluster \  
--replica-configuration \  
    ReplicaCount=1
```

Untuk Windows:

```
aws memorydb update-cluster ^  
--cluster-name my-cluster ^  
--replica-configuration ^  
    ReplicaCount=1 ^
```

Ini akan mengembalikan respon JSON berikut:

```
{  
  "Cluster": {  
    "Name": "my-cluster",  
    "Status": "updating",  
    "NumberOfShards": 1,  
    "ClusterEndpoint": {  
      "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-east-1.amazonaws.com",  
      "Port": 6379  
    },  
    "NodeType": "db.r6g.large",  
    "EngineVersion": "6.2",  
    "EnginePatchVersion": "6.2.6",  
    "ParameterGroupName": "default.memorydb-redis6",  
    "ParameterGroupStatus": "in-sync",  
    "SubnetGroupName": "my-sg",  
    "TLSEnabled": true,  
    "ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",  
    "SnapshotRetentionLimit": 0,  
    "MaintenanceWindow": "wed:03:00-wed:04:00",  
    "SnapshotWindow": "04:30-05:30",  
    "DataTiering": "false",  
    "AutoMinorVersionUpgrade": true  
  }  
}
```

Untuk melihat rincian klaster yang diperbarui setelah statusnya berubah dari memperbaiki ke tersedia, gunakan perintah berikut:

Untuk Linux, macOS, atau Unix:

```
aws memorydb describe-clusters \  
  --cluster-name my-cluster  
  --show-shard-details
```

Untuk Windows:

```
aws memorydb describe-clusters ^  
  --cluster-name my-cluster  
  --show-shard-details
```

Ini akan mengembalikan respon JSON berikut:

```
{  
  "Clusters": [  
    {  
      "Name": "my-cluster",  
      "Status": "available",  
      "NumberOfShards": 1,  
      "Shards": [  
        {  
          "Name": "0001",  
          "Status": "available",  
          "Slots": "0-16383",  
          "Nodes": [  
            {  
              "Name": "my-cluster-0001-001",  
              "Status": "available",  
              "AvailabilityZone": "us-east-1a",  
              "CreateTime": "2021-08-21T20:22:12.405000-07:00",  
              "Endpoint": {  
                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-east-1.amazonaws.com",  
                "Port": 6379  
              }  
            },  
            {  
              "Name": "my-cluster-0001-002",  
              "Status": "available",  
              "AvailabilityZone": "us-east-1b",  
              "CreateTime": "2021-08-21T20:22:12.405000-07:00",  
              "Endpoint": {
```



```

        "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
        "Port": 6379
    }
}
],
    "NumberOfNodes": 2
}
],
    "ClusterEndpoint": {
        "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
        "Port": 6379
    },
    "NodeType": "db.r6g.large",
    "EngineVersion": "6.2",
    "EnginePatchVersion": "6.2.6",
    "ParameterGroupName": "default.memorydb-redis6",
    "ParameterGroupStatus": "in-sync",
    "SubnetGroupName": "my-sg",
    "TLSEnabled": true,
    "ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",
    "SnapshotRetentionLimit": 0,
    "MaintenanceWindow": "wed:03:00-wed:04:00",
    "SnapshotWindow": "04:30-05:30",
    "ACLName": "my-acl",
    "DataTiering": "false",
    "AutoMinorVersionUpgrade": true
}
]
}

```

Untuk informasi lebih lanjut tentang mengurangi jumlah replika menggunakan CLI, lihat [update-cluster](#) di Command Reference. AWS CLI

Menggunakan API MemoryDB

Untuk mengurangi jumlah replika di dalam kluster MemoryDB, gunakan `UpdateCluster` tindakan dengan parameter berikut:

- `ClusterName` – Diperlukan. Mengidentifikasi kluster yang ingin dikurangi jumlah replika di dalamnya.
- `ReplicaConfiguration` – Diperlukan. Memungkinkan Anda mengatur jumlah replika.

`ReplicaCount`- Mengatur properti ini untuk menentukan jumlah simpul replika yang Anda inginkan.

Example

Contoh berikut menggunakan `ReplicaCount` untuk mengurangi jumlah replika di dalam kluster `sample-cluster` menjadi satu. Saat contoh ini selesai, terdapat satu replika di setiap serpihan. Nomor ini berlaku apakah ini adalah cluster MemoryDB dengan pecahan tunggal atau cluster MemoryDB dengan beberapa pecahan.

```
https://memory-db.us-east-1.amazonaws.com/  
?Action=UpdateCluster  
&ReplicaConfiguration.ReplicaCount=1  
&ClusterName=sample-cluster  
&Version=2021-01-01  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20210802T192317Z  
&X-Amz-Credential=<credential>
```

Untuk informasi lain tentang menurunkan jumlah replika menggunakan API, lihat. [UpdateCluster](#)

Melakukan snapshot dan pemulihan

Cluster MemoryDB secara otomatis mencadangkan data ke log transaksional multi-AZ, tetapi Anda dapat memilih untuk membuat point-in-time snapshot cluster baik secara berkala maupun sesuai permintaan. Snapshot ini dapat digunakan untuk membuat ulang cluster pada titik sebelumnya atau untuk menyemai cluster baru. Snapshot terdiri dari metadata cluster, bersama dengan semua data di cluster. Semua snapshot ditulis ke Amazon Simple Storage Service (Amazon S3), yang menyediakan penyimpanan tahan lama. Kapan saja, Anda dapat memulihkan data Anda dengan membuat cluster MemoryDB baru dan mengisinya dengan data dari snapshot. Dengan MemoryDB, Anda dapat mengelola snapshot menggunakan API, AWS Command Line Interface (AWS CLI) AWS Management Console, dan MemoryDB.

Topik

- [Kendala snapshot](#)
- [Biaya snapshot](#)

- [Menjadwalkan snapshot otomatis](#)
- [Membuat snapshot manual](#)
- [Membuat snapshot akhir](#)
- [Menggambarkan snapshot](#)
- [Menyalin snapshot](#)
- [Mengekspor snapshot](#)
- [Memulihkan dari snapshot](#)
- [Menyemai cluster baru dengan snapshot yang dibuat secara eksternal](#)
- [Menandai snapshot](#)
- [Menghapus snapshot](#)

Kendala snapshot

Pertimbangkan kendala berikut saat merencanakan atau membuat snapshot:

- Untuk cluster MemoryDB, snapshot dan restore tersedia untuk semua jenis node yang didukung.
- Selama periode 24 jam yang berdekatan, Anda dapat membuat tidak lebih dari 20 snapshot manual per cluster.
- MemoryDB hanya mendukung pengambilan snapshot di tingkat cluster. MemoryDB tidak mendukung pengambilan snapshot di level shard atau node.
- Selama proses snapshot, Anda tidak dapat menjalankan operasi API atau CLI lainnya di cluster.
- Jika Anda menghapus cluster dan meminta snapshot akhir, MemoryDB selalu mengambil snapshot dari node utama. Ini memastikan bahwa Anda menangkap data terbaru sebelum cluster dihapus.

Biaya snapshot

Menggunakan MemoryDB, Anda dapat menyimpan satu snapshot untuk setiap cluster MemoryDB aktif secara gratis. Ruang penyimpanan untuk snapshot tambahan dikenakan biaya sebesar \$0,085/GB per bulan untuk semua Wilayah. AWS Tidak ada biaya transfer data untuk membuat snapshot, atau untuk memulihkan data dari snapshot ke cluster MemoryDB.

Menjadwalkan snapshot otomatis

Untuk cluster MemoryDB apa pun, Anda dapat mengaktifkan snapshot otomatis. Saat snapshot otomatis diaktifkan, MemoryDB membuat snapshot cluster setiap hari. Tidak ada dampak pada klaster dan perubahan itu terjadi seketika. Untuk informasi selengkapnya, lihat [Memulihkan dari snapshot](#).

Saat Anda menjadwalkan snapshot otomatis, Anda harus merencanakan pengaturan berikut:

- **Jendela snapshot** — Periode selama setiap hari ketika MemoryDB mulai membuat snapshot. Panjang minimum untuk jendela snapshot adalah 60 menit. Anda dapat mengatur jendela snapshot kapan saja ketika itu paling nyaman bagi Anda, atau untuk waktu hari yang menghindari melakukan snapshot selama periode pemanfaatan yang sangat tinggi.

Jika Anda tidak menentukan jendela snapshot, MemoryDB menetapkannya secara otomatis.

- **Batas retensi snapshot** - Jumlah hari snapshot dipertahankan di Amazon S3. Misalnya, jika Anda menetapkan batas retensi ke 5, maka snapshot yang diambil hari ini dipertahankan selama 5 hari. Ketika batas retensi berakhir, snapshot secara otomatis dihapus.

Batas retensi snapshot maksimum adalah 35 hari. Jika batas retensi snapshot disetel ke 0, snapshot otomatis dinonaktifkan untuk cluster. Data MemoryDB masih sepenuhnya tahan lama bahkan dengan snapshotting otomatis dinonaktifkan.

Anda dapat mengaktifkan atau menonaktifkan snapshot otomatis saat membuat cluster MemoryDB menggunakan konsol MemoryDB, atau API MemoryDB. AWS CLI Anda dapat mengaktifkan snapshot otomatis saat membuat cluster MemoryDB dengan mencentang kotak Aktifkan Pencadangan Otomatis di bagian Snapshots. Untuk informasi selengkapnya, lihat [Membuat cluster MemoryDB](#).

Membuat snapshot manual

Selain snapshot otomatis, Anda dapat membuat snapshot manual kapan saja. Tidak seperti snapshot otomatis, yang secara otomatis dihapus setelah periode retensi tertentu, snapshot manual tidak memiliki periode retensi setelah itu dihapus secara otomatis. Anda harus menghapus snapshot manual secara manual. Bahkan jika Anda menghapus cluster atau node, snapshot manual apa pun dari cluster atau node tersebut tetap dipertahankan. Jika Anda tidak lagi ingin menyimpan snapshot manual, Anda harus menghapusnya sendiri secara eksplisit.

Snapshot manual berguna untuk pengujian dan pengarsipan. Sebagai contoh, misalkan Anda sudah mengembangkan sekumpulan data garis dasar untuk tujuan pengujian. Anda dapat membuat snapshot manual dari data dan mengembalikannya kapan pun Anda mau. Setelah menguji aplikasi yang memodifikasi data, Anda dapat mengatur ulang data dengan membuat cluster baru dan memulihkan dari snapshot dasar Anda. Ketika cluster siap, Anda dapat menguji aplikasi Anda terhadap data dasar lagi-dan ulangi proses ini sesering yang diperlukan.

Selain langsung membuat snapshot manual, Anda dapat membuat snapshot manual dengan salah satu cara berikut:

- [Menyalin snapshot](#)— Tidak masalah apakah snapshot sumber dibuat secara otomatis atau manual.
- [Membuat snapshot akhir](#)— Buat snapshot segera sebelum menghapus cluster.

Topik penting lainnya

- [Kendala snapshot](#)
- [Biaya snapshot](#)

Anda dapat membuat snapshot manual dari sebuah node menggunakan API AWS Management Console, the AWS CLI, atau MemoryDB.

Membuat snapshot manual (Konsol)

Untuk membuat snapshot dari cluster (konsol)

1. [Masuk ke AWS Management Console dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/.](https://console.aws.amazon.com/memorydb/)

2. dari panel navigasi kiri, pilih Clusters.

Layar cluster MemoryDB muncul.

3. pilih tombol radio di sebelah kiri nama cluster MemoryDB yang ingin Anda cadangkan.
4. Pilih Tindakan dan kemudian Ambil snapshot.
5. Di jendela Snapshot, ketik nama untuk snapshot Anda di Nama Snapshot kotak. Kami menyarankan agar nama tersebut menunjukkan cluster mana yang dicadangkan dan tanggal serta waktu snapshot dibuat.

Batasan penamaan klaster adalah sebagai berikut:

- Harus berisi 1–40 karakter alfanumerik atau tanda hubung.
 - Harus diawali dengan huruf.
 - Tidak boleh berisi dua tanda hubung berurutan.
 - Tidak boleh diakhiri dengan tanda hubung.
6. Di bawah Enkripsi, pilih apakah akan menggunakan kunci enkripsi default atau kunci yang dikelola pelanggan. Untuk informasi selengkapnya, lihat [Enkripsi dalam transit \(TLS\) di MemoryDB](#).
 7. Di bawah Tag, tambahkan tag secara opsional untuk mencari dan memfilter snapshot Anda atau melacak biaya Anda AWS .
 8. Pilih Ambil snapshot.

Status klaster berubah menjadi snapshotting. Ketika status kembali tersedia, snapshot selesai.

Membuat snapshot manual (AWS CLI)

Untuk membuat snapshot manual cluster menggunakan AWS CLI, gunakan `create-snapshot` AWS CLI operasi dengan parameter berikut:

- `--cluster-name`— Nama cluster MemoryDB untuk digunakan sebagai sumber untuk snapshot. Gunakan parameter ini saat membuat cadangan cluster MemoryDB.

Batasan penamaan klaster adalah sebagai berikut:

- Harus berisi 1–40 karakter alfanumerik atau tanda hubung.
- Harus diawali dengan huruf.
- Tidak boleh berisi dua tanda hubung berurutan.

- Tidak boleh diakhiri dengan tanda hubung.
- `--snapshot-name` – Nama snapshot yang akan dibuat.

Topik terkait

Untuk informasi selengkapnya, lihat `create-snapshot` dalam AWS CLI Referensi Perintah.

Membuat snapshot manual (MemoryDB API)

Untuk membuat snapshot manual cluster menggunakan API MemoryDB, gunakan operasi API `CreateSnapshot` MemoryDB dengan parameter berikut:

- `ClusterName`— Nama cluster MemoryDB untuk digunakan sebagai sumber untuk snapshot. Gunakan parameter ini saat membuat cadangan cluster MemoryDB.

Batasan penamaan klaster adalah sebagai berikut:

- Harus berisi 1–40 karakter alfanumerik atau tanda hubung.
- Harus diawali dengan huruf.
- Tidak boleh berisi dua tanda hubung berurutan.
- Tidak boleh diakhiri dengan tanda hubung.
- `SnapshotName` – Nama snapshot yang akan dibuat.

Topik terkait

Untuk informasi lebih lanjut, lihat [CreateSnapshot](#).

Membuat snapshot akhir

Anda dapat membuat snapshot akhir menggunakan konsol MemoryDB, API AWS CLI, atau MemoryDB.

Membuat snapshot akhir (Konsol)

Anda dapat membuat snapshot akhir saat Anda menghapus cluster MemoryDB menggunakan konsol MemoryDB.

Untuk membuat snapshot akhir saat menghapus cluster MemoryDB, pada halaman hapus, pilih Ya dan beri nama snapshot di. [Langkah 4: Menghapus cluster](#)

Membuat snapshot akhir (AWS CLI)

Anda dapat membuat snapshot akhir saat menghapus cluster MemoryDB menggunakan file. AWS CLI

Saat menghapus cluster MemoryDB

Untuk membuat snapshot akhir saat menghapus cluster, gunakan `delete-cluster` AWS CLI operasi, dengan parameter berikut:

- `--cluster-name` – Nama klaster yang dihapus.
- `--final-snapshot-name`— Nama snapshot terakhir.

Kode berikut mengambil snapshot terakhir `bkup-20210515-final` saat menghapus cluster. `myCluster`

Untuk Linux, macOS, atau Unix:

```
aws memorydb delete-cluster \  
  --cluster-name myCluster \  
  --final-snapshot-name bkup-20210515-final
```

Untuk Windows:

```
aws memorydb delete-cluster ^  
  --cluster-name myCluster ^  
  --final-snapshot-name bkup-20210515-final
```


Untuk informasi selengkapnya, lihat [menghapus-cluster di Referensi AWS CLI Perintah](#).

Membuat snapshot akhir (MemoryDB API)

Anda dapat membuat snapshot akhir saat menghapus cluster MemoryDB menggunakan API MemoryDB.

Saat menghapus cluster MemoryDB

Untuk membuat snapshot akhir, gunakan operasi API DeleteCluster MemoryDB dengan parameter berikut.

- `ClusterName` – Nama klaster yang dihapus.
- `FinalSnapshotName`— Nama snapshot.

Operasi API MemoryDB berikut membuat snapshot `bkup-20210515-final` saat menghapus cluster `myCluster`

```
https://memory-db.us-east-1.amazonaws.com/  
?Action=DeleteCluster  
&ClusterName=myCluster  
&FinalSnapshotName=bkup-20210515-final  
&Version=2021-01-01  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20210515T192317Z  
&X-Amz-Credential=<credential>
```

Untuk informasi lebih lanjut, lihat [DeleteCluster](#).

Menggambarkan snapshot

Prosedur berikut menunjukkan cara menampilkan daftar snapshot Anda. Jika mau, Anda juga dapat melihat detail snapshot tertentu.

Menjelaskan snapshot (Konsol)

Untuk menampilkan snapshot menggunakan AWS Management Console

1. Masuk ke konsol
2. dari panel navigasi kiri, pilih Snapshots.
3. Gunakan pencarian untuk memfilter secara manual, otomatis, atau semua snapshot.
4. Untuk melihat detail snapshot tertentu, pilih tombol radio di sebelah kiri nama snapshot. Pilih Tindakan dan kemudian Lihat detail.
5. Secara opsional, di halaman Lihat detail, Anda dapat melakukan tindakan snapshot tambahan seperti menyalin, memulihkan, atau menghapus. Anda juga dapat menambahkan tag ke snapshot

Menjelaskan snapshot (CLI AWS)

Untuk menampilkan daftar snapshot dan detail opsional tentang snapshot tertentu, gunakan operasi `CLIdescribe-snapshots`.

Contoh

Operasi berikut menggunakan parameter `--max-results` untuk mencantumkan hingga 20 snapshot yang terkait dengan akun Anda. Menghilangkan `--max-results` daftar parameter hingga 50 snapshot.

```
aws memorydb describe-snapshots --max-results 20
```

Operasi berikut menggunakan parameter `--cluster-name` untuk daftar hanya snapshot yang terkait dengan `clustermy-cluster`.

```
aws memorydb describe-snapshots --cluster-name my-cluster
```

Operasi berikut menggunakan parameter `--snapshot-name` untuk menampilkan detail `snapshotmy-snapshot`.

```
aws memorydb describe-snapshots --snapshot-name my-snapshot
```

Untuk informasi selengkapnya, lihat [deskripsi-snapshot](#).

Menjelaskan snapshot (MemoryDB API)

Untuk menampilkan daftar snapshot, gunakan DescribeSnapshots operasi.

Contoh

Operasi berikut menggunakan parameter MaxResults untuk mencantumkan hingga 20 snapshot yang terkait dengan akun Anda. Menghilangkan MaxResults daftar parameter hingga 50 snapshot.

```
https://memory-db.us-east-1.amazonaws.com/  
?Action=DescribeSnapshots  
&MaxResults=20  
&SignatureMethod=HmacSHA256  
&SignatureVersion=4  
&Timestamp=20210801T220302Z  
&Version=2021-01-01  
&X-Amz-Algorithm=Amazon4-HMAC-SHA256  
&X-Amz-Date=20210801T220302Z  
&X-Amz-SignedHeaders=Host  
&X-Amz-Expires=20210801T220302Z  
&X-Amz-Credential=<credential>  
&X-Amz-Signature=<signature>
```

Operasi berikut menggunakan parameter ClusterName untuk daftar semua snapshot yang terkait dengan clusterMyCluster.

```
https://memory-db.us-east-1.amazonaws.com/  
?Action=DescribeSnapshots  
&ClusterName=MyCluster  
&SignatureMethod=HmacSHA256  
&SignatureVersion=4  
&Timestamp=20210801T220302Z  
&Version=2021-01-01  
&X-Amz-Algorithm=Amazon4-HMAC-SHA256  
&X-Amz-Date=20210801T220302Z  
&X-Amz-SignedHeaders=Host  
&X-Amz-Expires=20210801T220302Z  
&X-Amz-Credential=<credential>
```

```
&X-Amz-Signature=<signature>
```

Operasi berikut menggunakan parameter `SnapshotName` untuk menampilkan detail `snapshotMyBackup`.

```
https://memory-db.us-east-1.amazonaws.com/  
?Action=DescribeSnapshots  
&SignatureMethod=HmacSHA256  
&SignatureVersion=4  
&SnapshotName=MyBackup  
&Timestamp=20210801T220302Z  
&Version=2021-01-01  
&X-Amz-Algorithm=Amazon4-HMAC-SHA256  
&X-Amz-Date=20210801T220302Z  
&X-Amz-SignedHeaders=Host  
&X-Amz-Expires=20210801T220302Z  
&X-Amz-Credential=<credential>  
&X-Amz-Signature=<signature>
```

Untuk informasi lebih lanjut, lihat [DescribeSnapshots](#).

Menyalin snapshot

Anda dapat membuat salinan snapshot apa pun, apakah itu dibuat secara otomatis atau manual. Saat menyalin snapshot, kunci enkripsi KMS yang sama dengan sumber digunakan untuk target kecuali secara khusus diganti. Anda juga dapat mengekspor snapshot Anda sehingga Anda dapat mengaksesnya dari luar MemoryDB. Untuk panduan tentang mengekspor snapshot Anda, lihat [Mengekspor snapshot](#)

[Mengekspor snapshot](#)

Prosedur berikut menunjukkan kepada Anda cara menyalin snapshot.

Menyalin snapshot (Konsol)

Untuk menyalin snapshot (konsol)

1. [Masuk ke AWS Management Console dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/.](https://console.aws.amazon.com/memorydb/)
2. Untuk melihat daftar snapshot Anda, dari panel navigasi kiri pilih Snapshots.
3. Dari daftar snapshot, pilih tombol radio di sebelah kiri nama snapshot yang ingin Anda salin.
4. Pilih Tindakan dan kemudian pilih Salin.
5. Di halaman Salin snapshot, lakukan hal berikut:
 - a. Di kotak Nama snapshot baru, ketikkan nama untuk snapshot baru Anda.
 - b. Biarkan kotak opsional Bucket S3 Target kosong. Bidang ini hanya boleh digunakan untuk mengekspor snapshot Anda dan memerlukan izin S3 khusus. Untuk informasi tentang mengekspor snapshot, lihat [Mengekspor snapshot](#)
 - c. Pilih apakah akan menggunakan kunci AWS KMS enkripsi default atau menggunakan kunci khusus. Untuk informasi selengkapnya, lihat [Enkripsi dalam transit \(TLS\) di MemoryDB](#).
 - d. Secara opsional, Anda juga dapat menambahkan tag ke salinan snapshot.
 - e. Pilih Salin.

Menyalin snapshot (CLI AWS)

Untuk menyalin snapshot, gunakan copy-snapshot operasi.

Parameter

- `--source-snapshot-name`— Nama snapshot yang akan disalin.

- `--target-snapshot-name`— Nama salinan snapshot.
- `--target-bucket`— Dicadangkan untuk mengeksplor snapshot. Jangan gunakan parameter ini saat membuat salinan snapshot. Untuk informasi selengkapnya, lihat [Mengeksplor snapshot](#).

Contoh berikut membuat salinan snapshot otomatis.

Untuk Linux, macOS, atau Unix:

```
aws memorydb copy-snapshot \  
  --source-snapshot-name automatic.my-primary-2021-03-27-03-15 \  
  --target-snapshot-name my-snapshot-copy
```

Untuk Windows:

```
aws memorydb copy-snapshot ^  
  --source-snapshot-name automatic.my-primary-2021-03-27-03-15 ^  
  --target-snapshot-name my-snapshot-copy
```

Untuk informasi selengkapnya, lihat [copy-snapshot](#).

Menyalin snapshot (MemoryDB API)

Untuk menyalin snapshot, gunakan `copy-snapshot` operasi dengan parameter berikut:

Parameter

- `SourceSnapshotName`— Nama snapshot yang akan disalin.
- `TargetSnapshotName`— Nama salinan snapshot.
- `TargetBucket`— Dicadangkan untuk mengeksplor snapshot. Jangan gunakan parameter ini saat membuat salinan snapshot. Untuk informasi selengkapnya, lihat [Mengeksplor snapshot](#).

Contoh berikut membuat salinan snapshot otomatis.

Example

```
https://memory-db.us-east-1.amazonaws.com/  
  ?Action=CopySnapshot  
  &SourceSnapshotName=automatic.my-primary-2021-03-27-03-15  
  &TargetSnapshotName=my-snapshot-copy
```

```
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20210801T220302Z
&Version=2021-01-01
&X-Amz-Algorithm=Amazon4-HMAC-SHA256
&X-Amz-Date=20210801T220302Z
&X-Amz-SignedHeaders=Host
&X-Amz-Expires=20210801T220302Z
&X-Amz-Credential=<credential>
&X-Amz-Signature=<signature>
```

Untuk informasi lebih lanjut, lihat [CopySnapshot](#).

Mengekspor snapshot

MemoryDB mendukung ekspor snapshot MemoryDB Anda ke bucket Amazon Simple Storage Service (Amazon S3), yang memberi Anda akses ke sana dari luar MemoryDB. Snapshot MemoryDB yang diekspor sepenuhnya sesuai dengan Redis OSS open-source dan dapat dimuat dengan versi atau perangkat Redis OSS yang sesuai. Anda dapat mengekspor snapshot menggunakan konsol MemoryDB, API AWS CLI, atau MemoryDB.

Mengekspor snapshot dapat membantu jika Anda perlu meluncurkan cluster di Wilayah lain AWS . Anda dapat mengekspor data Anda dalam satu AWS Wilayah, menyalin file.rdb ke AWS Wilayah baru, dan kemudian menggunakan file.rdb itu untuk menyemai cluster baru alih-alih menunggu klaster baru diisi melalui penggunaan. Untuk informasi tentang melakukan seeding klaster baru, lihat [Menyemai cluster baru dengan snapshot yang dibuat secara eksternal](#). Alasan lain Anda sebaiknya mengekspor data dari klaster Anda adalah untuk menggunakan file .rdb untuk pemrosesan offline.

Important

- Snapshot MemoryDB dan bucket Amazon S3 yang ingin Anda salin harus berada di Wilayah yang sama. AWS

Meskipun snapshot yang disalin ke bucket Amazon S3 dienkripsi, kami sangat menyarankan agar Anda tidak memberi orang lain akses ke bucket Amazon S3 tempat Anda ingin menyimpan snapshot Anda.

- Mengekspor snapshot ke Amazon S3 tidak didukung untuk cluster yang menggunakan tiering data. Untuk informasi selengkapnya, lihat [Tingkatan data](#).

Sebelum dapat mengekspor snapshot ke bucket Amazon S3, Anda harus memiliki bucket Amazon S3 di Wilayah yang AWS sama dengan snapshot. Berikan akses MemoryDB ke ember. Dua langkah pertama menunjukkan cara melakukannya.

Warning

Skenario berikut mengungkapkan data Anda dengan cara yang mungkin tidak Anda inginkan:

- Ketika orang lain memiliki akses ke bucket Amazon S3 tempat Anda mengekspor snapshot Anda.

Untuk mengontrol akses ke snapshot Anda, hanya izinkan akses ke bucket Amazon S3 kepada mereka yang ingin mengakses data Anda. Untuk informasi tentang mengelola akses ke bucket Amazon S3, lihat [Mengelola akses di Panduan Pengembang Amazon S3](#).

- Ketika orang lain memiliki izin untuk menggunakan operasi CopySnapshot API.

Pengguna atau grup yang memiliki izin untuk menggunakan operasi CopySnapshot API dapat membuat bucket Amazon S3 mereka sendiri dan menyalin snapshot ke mereka. Untuk mengontrol akses ke snapshot Anda, gunakan kebijakan AWS Identity and Access Management (IAM) untuk mengontrol siapa yang memiliki kemampuan untuk menggunakan API. CopySnapshot Untuk informasi selengkapnya tentang penggunaan IAM untuk mengontrol penggunaan operasi API MemoryDB, lihat [Manajemen identitas dan akses di MemoryDB](#) di Panduan Pengguna MemoryDB.

Topik

- [Langkah 1: Buat bucket Amazon S3.](#)
- [Langkah 2: Berikan akses MemoryDB ke bucket Amazon S3 Anda](#)
- [Langkah 3: Ekspor snapshot MemoryDB](#)

Langkah 1: Buat bucket Amazon S3.

Prosedur berikut menggunakan konsol Amazon S3 untuk membuat bucket Amazon S3 tempat Anda mengekspor dan menyimpan snapshot MemoryDB Anda.

Untuk membuat bucket Amazon S3

1. [Masuk ke AWS Management Console dan buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.](https://console.aws.amazon.com/s3/)
2. Pilih Buat Bucket.
3. Di Buat Bucket - Pilih Nama Bucket dan Wilayah, lakukan hal berikut:
 - a. Di Nama Bucket, ketikkan nama untuk bucket Amazon S3 Anda.
 - b. Dari daftar Wilayah, pilih AWS Wilayah untuk bucket Amazon S3 Anda. AWS Wilayah ini harus AWS Wilayah yang sama dengan snapshot MemoryDB yang ingin Anda ekspor.
 - c. Pilih Buat.

Untuk informasi selengkapnya tentang cara membuat bucket Amazon S3, lihat [Membuat bucket](#) dalam Panduan Pengguna Amazon Simple Storage Service.

Langkah 2: Berikan akses MemoryDB ke bucket Amazon S3 Anda

AWS Wilayah yang diperkenalkan sebelum 20 Maret 2019, diaktifkan secara default. Anda dapat mulai bekerja di AWS Wilayah ini segera. Wilayah yang diperkenalkan setelah 20 Maret 2019 dinonaktifkan secara default. Anda harus mengaktifkan, atau ikut serta, ke Wilayah ini sebelum Anda dapat menggunakannya, seperti yang dijelaskan dalam [Mengelola AWS wilayah](#).

Berikan akses MemoryDB ke Bucket S3 Anda di suatu Wilayah AWS

Untuk membuat izin yang tepat pada bucket Amazon S3 di AWS Wilayah, lakukan langkah-langkah berikut.

Untuk memberikan akses MemoryDB ke bucket S3

1. [Masuk ke AWS Management Console dan buka konsol Amazon S3 di https://console.aws.amazon.com/s3/](https://console.aws.amazon.com/s3/).
2. Pilih nama bucket Amazon S3 yang ingin Anda salin snapshot. Bucket tersebut harus berupa bucket S3 yang Anda buat di [Langkah 1: Buat bucket Amazon S3](#).
3. Pilih tab Izin dan di bawah Izin, pilih Kebijakan Bucket.
4. Perbarui kebijakan untuk memberikan izin yang diperlukan MemoryDB untuk melakukan operasi:
 - Tambahkan ["Service" : "*region-full-name*.memorydb-snapshot.amazonaws.com"] ke Principal.
 - Tambahkan izin berikut yang diperlukan untuk mengekspor snapshot ke bucket Amazon S3.
 - "s3:PutObject"
 - "s3:GetObject"
 - "s3:ListBucket"
 - "s3:GetBucketAcl"
 - "s3:ListMultipartUploadParts"
 - "s3:ListBucketMultipartUploads"

Berikut adalah contoh tampilan kebijakan yang sudah diperbarui.

```
{
```

```

"Version": "2012-10-17",
"Id": "Policy15397346",
"Statement": [
  {
    "Sid": "Stmt15399483",
    "Effect": "Allow",
    "Principal": {
      "Service": "aws-region.memorydb-snapshot.amazonaws.com"
    },
    "Action": [
      "s3:PutObject",
      "s3:GetObject",
      "s3:ListBucket",
      "s3:GetBucketAcl",
      "s3:ListMultipartUploadParts",
      "s3:ListBucketMultipartUploads"
    ],
    "Resource": [
      "arn:aws:s3:::example-bucket",
      "arn:aws:s3:::example-bucket/*"
    ]
  }
]
}

```

Langkah 3: Ekspor snapshot MemoryDB

Sekarang Anda telah membuat bucket S3 Anda dan memberikan izin MemoryDB untuk mengaksesnya. Ubah Kepemilikan Objek S3 menjadi ACL yang diaktifkan - Pemilik bucket lebih disukai. Selanjutnya, Anda dapat menggunakan konsol MemoryDB, AWS CLI, atau API MemoryDB untuk mengekspor snapshot Anda ke sana. Berikut ini mengasumsikan bahwa Anda memiliki izin IAM khusus S3 tambahan berikut.

```

{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "s3:GetBucketLocation",
      "s3:ListAllMyBuckets",
      "s3:PutObject",
      "s3:GetObject",

```

```
"s3:DeleteObject",
"s3:ListBucket"
],
"Resource": "arn:aws:s3:::*"
}]
}
```

Mengekspor snapshot MemoryDB (Konsol)

Proses berikut menggunakan konsol MemoryDB untuk mengekspor snapshot ke bucket Amazon S3 sehingga Anda dapat mengaksesnya dari luar MemoryDB. Bucket Amazon S3 harus berada di AWS Wilayah yang sama dengan snapshot MemoryDB.

Untuk mengekspor snapshot MemoryDB ke bucket Amazon S3

1. [Masuk ke AWS Management Console dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/.](https://console.aws.amazon.com/memorydb/)
2. Untuk melihat daftar snapshot Anda, dari panel navigasi kiri pilih Snapshots.
3. Dari daftar snapshot, pilih tombol radio di sebelah kiri nama snapshot yang ingin Anda ekspor.
4. Pilih Salin.
5. Di Buat Salinan Cadangan?, lakukan hal berikut:
 - a. Di kotak nama snapshot baru, ketikkan nama untuk snapshot baru Anda.

Nama tersebut harus berisi antara 1 dan 1.000 karakter serta diekode dengan UTF-8.

MemoryDB menambahkan pengenal pecahan dan nilai yang .rdb Anda masukkan di sini. Misalnya, jika Anda masukmy-exported-snapshot, MemoryDB menciptakan. my-exported-snapshot-0001.rdb

- b. Dari daftar Lokasi Target S3, pilih nama bucket Amazon S3 yang ingin Anda salin snapshot (bucket yang Anda buat). [Langkah 1: Buat bucket Amazon S3.](#)

Lokasi Target S3 harus berupa bucket Amazon S3 di Wilayah snapshot AWS dengan izin berikut agar proses ekspor berhasil.

- Akses objek – Baca dan Tulis.
- Akses izin – Baca.

Untuk informasi selengkapnya, lihat [Langkah 2: Berikan akses MemoryDB ke bucket Amazon S3 Anda](#).

- c. Pilih Salin.

Note

Jika bucket S3 Anda tidak memiliki izin yang diperlukan untuk MemoryDB untuk mengekspor snapshot ke sana, Anda menerima salah satu pesan galat berikut. Kembali ke [Langkah 2: Berikan akses MemoryDB ke bucket Amazon S3 Anda](#) untuk menambahkan izin yang ditentukan dan coba lagi mengekspor snapshot Anda.

- MemoryDB belum diberikan izin BACA %s pada Bucket S3.

Solusi: Tambahkan izin Baca pada bucket.

- MemoryDB belum diberikan izin WRITE %s pada Bucket S3.

Solusi: Tambahkan izin Tulis pada bucket.

- MemoryDB belum diberikan izin READ_ACP %s pada Bucket S3.

Solusi: Tambahkan Baca untuk akses Izin pada bucket.

Jika Anda ingin menyalin snapshot Anda ke AWS Wilayah lain, gunakan Amazon S3 untuk menyalinnya. Untuk informasi selengkapnya, lihat [Menyalin objek](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Mengekspor snapshot MemoryDB (CLI)AWS

Ekspor snapshot ke bucket Amazon S3 menggunakan operasi copy-snapshot CLI dengan parameter berikut:

Parameter

- `--source-snapshot-name`— Nama snapshot yang akan disalin.
- `--target-snapshot-name`— Nama salinan snapshot.

Nama tersebut harus berisi antara 1 dan 1.000 karakter serta diekode dengan UTF-8.

MemoryDB menambahkan pengenal pecahan dan nilai yang `.rdb` Anda masukkan di sini.

Misalnya, jika Anda masuk `my-exported-snapshot`, MemoryDB menciptakan `my-exported-snapshot-0001.rdb`

- `--target-bucket`— Nama bucket Amazon S3 tempat Anda ingin mengekspor snapshot. Salinan snapshot dibuat di ember yang ditentukan.

`--target-bucket` Harus berupa bucket Amazon S3 di AWS Wilayah snapshot dengan izin berikut agar proses ekspor berhasil.

- Akses objek – Baca dan Tulis.
- Akses izin – Baca.

Untuk informasi selengkapnya, lihat [Langkah 2: Berikan akses MemoryDB ke bucket Amazon S3 Anda](#).

Operasi berikut menyalin snapshot ke `my-s3-bucket`.

Untuk Linux, macOS, atau Unix:

```
aws memorydb copy-snapshot \  
  --source-snapshot-name automatic.my-primary-2021-06-27-03-15 \  
  --target-snapshot-name my-exported-snapshot \  
  --target-bucket my-s3-bucket
```

Untuk Windows:

```
aws memorydb copy-snapshot ^  
  --source-snapshot-name automatic.my-primary-2021-06-27-03-15 ^  
  --target-snapshot-name my-exported-snapshot ^  
  --target-bucket my-s3-bucket
```

Note

Jika bucket S3 Anda tidak memiliki izin yang diperlukan untuk MemoryDB untuk mengekspor snapshot ke sana, Anda menerima salah satu pesan galat berikut. Kembali ke [Langkah 2: Berikan akses MemoryDB ke bucket Amazon S3 Anda](#) untuk menambahkan izin yang ditentukan dan coba lagi mengekspor snapshot Anda.

- MemoryDB belum diberikan izin BACA %s pada Bucket S3.

Solusi: Tambahkan izin Baca pada bucket.

- MemoryDB belum diberikan izin WRITE %s pada Bucket S3.

Solusi: Tambahkan izin Tulis pada bucket.

- MemoryDB belum diberikan izin READ_ACP %s pada Bucket S3.

Solusi: Tambahkan Baca untuk akses Izin pada bucket.

Untuk informasi selengkapnya, lihat `copy-snapshot` dalam AWS CLI Referensi Perintah.

Jika Anda ingin menyalin snapshot Anda ke AWS Wilayah lain, gunakan salinan Amazon S3.

Untuk informasi selengkapnya, lihat [Menyalin objek](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Mengekspor snapshot MemoryDB (MemoryDB API)

Ekspor snapshot ke bucket Amazon S3 menggunakan operasi API dengan parameter `CopySnapshot` ini.

Parameter

- `SourceSnapshotName`— Nama snapshot yang akan disalin.
- `TargetSnapshotName`— Nama salinan snapshot.

Nama tersebut harus berisi antara 1 dan 1.000 karakter serta diekode dengan UTF-8.

MemoryDB menambahkan pengenal pecahan dan nilai yang `.rdb` Anda masukkan di sini.

Misalnya, jika Anda memasukkan `my-exported-snapshot`, Anda memperoleh `my-exported-snapshot-0001.rdb`.

- `TargetBucket`— Nama bucket Amazon S3 tempat Anda ingin mengekspor snapshot. Salinan snapshot dibuat di ember yang ditentukan.

`TargetBucket` harus berupa bucket Amazon S3 di AWS Wilayah snapshot dengan izin berikut agar proses ekspor berhasil.

- Akses objek – Baca dan Tulis.
- Akses izin – Baca.

Untuk informasi selengkapnya, lihat [Langkah 2: Berikan akses MemoryDB ke bucket Amazon S3 Anda](#).

Contoh berikut membuat salinan snapshot otomatis ke bucket Amazon my-s3-bucket S3.

Example

```
https://memory-db.us-east-1.amazonaws.com/  
?Action=CopySnapshot  
&SourceSnapshotName=automatic.my-primary-2021-06-27-03-15  
&TargetBucket=my-s3-bucket  
&TargetSnapshotName=my-snapshot-copy  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20210801T220302Z  
&Version=2021-01-01  
&X-Amz-Algorithm=Amazon4-HMAC-SHA256  
&X-Amz-Date=20210801T220302Z  
&X-Amz-SignedHeaders=Host  
&X-Amz-Expires=20210801T220302Z  
&X-Amz-Credential=<credential>  
&X-Amz-Signature=<signature>
```

Note

Jika bucket S3 Anda tidak memiliki izin yang diperlukan untuk MemoryDB untuk mengekspor snapshot ke sana, Anda menerima salah satu pesan galat berikut. Kembali ke [Langkah 2: Berikan akses MemoryDB ke bucket Amazon S3 Anda](#) untuk menambahkan izin yang ditentukan dan coba lagi mengekspor snapshot Anda.

- MemoryDB belum diberikan izin BACA %s pada Bucket S3.

Solusi: Tambahkan izin Baca pada bucket.

- MemoryDB belum diberikan izin WRITE %s pada Bucket S3.

Solusi: Tambahkan izin Tulis pada bucket.

- MemoryDB belum diberikan izin READ_ACP %s pada Bucket S3.

Solusi: Tambahkan Baca untuk akses Izin pada bucket.

Untuk informasi lebih lanjut, lihat [CopySnapshot](#).

Jika Anda ingin menyalin snapshot Anda ke AWS Wilayah lain, gunakan salinan Amazon S3 untuk menyalin snapshot yang diekspor ke bucket Amazon S3 di Wilayah lain. AWS Untuk informasi selengkapnya, lihat [Menyalin objek](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Memulihkan dari snapshot

Anda dapat mengembalikan data dari file snapshot MemoryDB atau ElastiCache (Redis OSS) .rdb ke cluster baru kapan saja.

Proses pemulihan MemoryDB mendukung hal-hal berikut:

- Bermigrasi dari satu atau lebih file snapshot .rdb yang Anda buat dari ElastiCache (Redis OSS) ke cluster MemoryDB.

File .rdb harus dimasukkan ke dalam S3 untuk melakukan pemulihan.

- Menentukan sejumlah pecahan di cluster baru yang berbeda dari jumlah pecahan di cluster yang digunakan untuk membuat file snapshot.
- Menentukan jenis node yang berbeda untuk cluster baru—lebih besar atau lebih kecil. Jika penskalaan ke tipe node yang lebih kecil, pastikan bahwa tipe node baru memiliki memori yang cukup untuk data Anda dan overhead Redis OSS.
- Mengkonfigurasi slot cluster MemoryDB baru secara berbeda dari pada cluster yang digunakan untuk membuat file snapshot.

Important

- Cluster MemoryDB tidak mendukung banyak database. Oleh karena itu, saat memulihkan ke MemoryDB pemulihan Anda gagal jika file.rdb mereferensikan lebih dari satu database.
- Anda tidak dapat mengembalikan snapshot dari cluster yang menggunakan tiering data (misalnya, tipe node r6gd) ke dalam cluster yang tidak menggunakan tiering data (misalnya, tipe node r6g).

Apakah Anda membuat perubahan saat memulihkan cluster dari snapshot diatur oleh pilihan yang Anda buat. Anda membuat pilihan ini di halaman Restore Cluster saat menggunakan konsol MemoryDB untuk memulihkan. Anda membuat pilihan ini dengan menetapkan nilai parameter saat menggunakan AWS CLI atau MemoryDB API untuk memulihkan.

Selama operasi pemulihan, MemoryDB membuat cluster baru, dan kemudian mengisinya dengan data dari file snapshot. Ketika proses ini selesai, cluster dihangatkan dan siap menerima permintaan.

⚠ Important

Sebelum Anda melanjutkan, pastikan Anda telah membuat snapshot dari cluster yang ingin Anda pulihkan. Untuk informasi selengkapnya, lihat [Membuat snapshot manual](#).

Jika Anda ingin memulihkan dari snapshot yang dibuat secara eksternal, lihat [Menyemai cluster baru dengan snapshot yang dibuat secara eksternal](#)

Prosedur berikut menunjukkan cara mengembalikan snapshot ke cluster baru menggunakan konsol MemoryDB, API AWS CLI, atau MemoryDB.

Memulihkan dari snapshot (Konsol)

Untuk mengembalikan snapshot ke cluster baru (konsol)

1. [Masuk ke AWS Management Console dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/](https://console.aws.amazon.com/memorydb/).
2. Pada panel navigasi, pilih Snapshots.
3. Dalam daftar snapshot, pilih tombol di sebelah nama nama snapshot yang ingin Anda pulihkan.
4. Pilih Tindakan dan kemudian pilih Pulihkan
5. Di bawah konfigurasi Cluster, masukkan yang berikut ini:
 - a. Nama cluster - Diperlukan. Nama klaster baru.
 - b. Deskripsi - Opsional. Deskripsi cluster baru.
6. Lengkapi bagian grup Subnet:
 - Untuk grup Subnet, buat grup subnet baru atau pilih yang sudah ada dari daftar yang tersedia yang ingin Anda terapkan ke cluster ini. Jika Anda membuat yang baru:
 - Masukkan Nama
 - Masukkan Deskripsi
 - Jika Anda mengaktifkan Multi-AZ, grup subnet harus berisi setidaknya dua subnet yang berada dalam availability zone yang berbeda. Untuk informasi selengkapnya, lihat [Subnet dan grup subnet](#).
 - Jika Anda membuat grup subnet baru dan tidak memiliki VPC yang ada, Anda akan diminta untuk membuat VPC. Untuk informasi lebih lanjut, lihat [Apa itu Amazon VPC?](#) di Panduan Pengguna Amazon VPC.

7. Lengkapi bagian Pengaturan Cluster:

- a. Untuk kompatibilitas versi Redis OSS, terima default. 6.0
- b. Untuk Port, terima port Redis OSS default 6379 atau, jika Anda memiliki alasan untuk menggunakan port yang berbeda, masukkan nomor port..
- c. Untuk grup Parameter, terima grup default .memorydb-redis6 parameter.

Grup parameter mengontrol parameter runtime dari klaster Anda. Untuk mengetahui informasi selengkapnya tentang grup parameter, lihat [Parameter spesifik Redis OSS](#).

- d. Untuk tipe Node, pilih nilai untuk tipe node (bersama dengan ukuran memori terkait) yang Anda inginkan.

Jika Anda memilih anggota keluarga tipe node r6gd, Anda akan secara otomatis mengaktifkan data-tiering di cluster Anda. Untuk informasi selengkapnya, lihat [Tingkatan data](#).

- e. Untuk Jumlah pecahan, pilih jumlah pecahan yang Anda inginkan untuk cluster ini.

Anda dapat mengubah jumlah pecahan di cluster Anda secara dinamis. Untuk informasi selengkapnya, lihat [Penskalaan cluster MemoryDB](#).

- f. Untuk Replika per serpihan, pilih jumlah simpul replika baca yang Anda inginkan dalam setiap serpihan.

Pembatasan berikut ada;

- Jika Multi-AZ diaktifkan, pastikan bahwa Anda memiliki setidaknya satu replika per serpihan.
- Replika akan berjumlah sama untuk setiap serpihan saat membuat klaster menggunakan konsol.

- g. Pilih Berikutnya


- h. Lengkapi bagian Pengaturan lanjutan:

- i. Untuk Grup keamanan, pilih grup keamanan yang Anda inginkan untuk klaster ini. Grup keamanan bertindak sebagai firewall untuk mengontrol akses jaringan ke klaster Anda. Anda dapat menggunakan grup keamanan default untuk VPC Anda atau membuat yang baru.

Untuk informasi selengkapnya tentang grup keamanan, lihat [Grup Keamanan untuk](#)

ii. Data dienkripsi dengan cara berikut:

- Enkripsi diam – Mengaktifkan enkripsi pada data yang disimpan di disk. Untuk informasi selengkapnya, lihat [Enkripsi Diam](#).

 Note

Anda memiliki opsi untuk menyediakan kunci enkripsi yang berbeda dengan memilih kunci AWS KMS yang Dikelola Pelanggan dan memilih kunci.

- Enkripsi bergerak – Mengaktifkan enkripsi data selama pengiriman. Pengaturan ini diaktifkan secara default. Untuk informasi selengkapnya, lihat [Enkripsi bergerak](#).

Jika Anda memilih tidak ada enkripsi, maka daftar kontrol Akses terbuka yang disebut “akses terbuka” akan dibuat dengan pengguna default. Untuk informasi selengkapnya, lihat [Mengautentikasi pengguna dengan Daftar Kontrol Akses \(ACL\)](#).

- iii. Untuk Snapshot secara opsional, tentukan periode retensi snapshot dan jendela snapshot. Secara default, Aktifkan snapshot otomatis dipilih.
- iv. Untuk jendela Pemeliharaan opsional menentukan jendela pemeliharaan. Jendela pemeliharaan adalah waktu, umumnya satu jam panjangnya, setiap minggu ketika MemoryDB menjadwalkan pemeliharaan sistem untuk cluster Anda. Anda dapat mengizinkan MemoryDB untuk memilih hari dan waktu untuk jendela pemeliharaan Anda (Tidak ada preferensi), atau Anda dapat memilih hari, waktu, dan durasi sendiri (Tentukan jendela pemeliharaan). Jika Anda memilih Tentukan periode pemeliharaan dari daftar, pilih Hari mulai, Waktu mulai, dan Durasi (dalam jam) untuk periode pemeliharaan. Semua waktu menggunakan zona waktu UTC.

Untuk informasi selengkapnya, lihat [Mengelola pemeliharaan](#).

- v. Untuk Pemberitahuan, pilih topik Amazon Simple Notification Service (Amazon SNS) yang sudah ada, atau pilih Input ARN Manual dan masukkan Nama Sumber Daya Amazon (ARN) topik. Amazon SNS memungkinkan Anda untuk mendorong pemberitahuan ke perangkat pintar yang terhubung ke Internet. Defaultnya adalah menonaktifkan notifikasi. Untuk informasi lebih lanjut, lihat <https://aws.amazon.com/sns/>.

- i. Untuk Tag, Anda dapat menerapkan tag secara opsional untuk mencari dan memfilter cluster Anda atau melacak biaya Anda AWS .

- j. Tinjau semua entri dan pilihan Anda, lalu lakukan koreksi yang diperlukan. Saat Anda siap, pilih **Buat klaster** untuk meluncurkan klaster Anda, atau **Batalkan** untuk membatalkan operasi.

Setelah status klaster Anda menjadi tersedia, Anda dapat memberi EC2 akses ke klaster tersebut, terhubung dengannya, dan mulai menggunakannya. Untuk informasi selengkapnya, lihat [Langkah 2: Otorisasi akses ke cluster](#) dan [Langkah 3: Connect ke cluster](#).

 **Important**

Setelah klaster Anda tersedia, Anda akan ditagih untuk setiap jam atau jam parsial saat klaster aktif, meskipun Anda tidak sedang aktif menggunakannya. Untuk menghentikan tagihan untuk klaster ini, Anda harus menghapusnya. Lihat [Langkah 4: Menghapus cluster](#).

Memulihkan dari snapshot (CLI AWS)

Saat menggunakan salah satu `create-cluster` operasi, pastikan untuk menyertakan parameter `--snapshot-name` atau `--snapshot-arns` menyemai cluster baru dengan data dari snapshot.

Untuk informasi selengkapnya, lihat berikut ini:

- [Membuat klaster AWS \(CLI\)](#) di Panduan Pengguna MemoryDB.
- [buat-cluster](#) di Command Reference. AWS CLI

Memulihkan dari snapshot (MemoryDB API)

Anda dapat memulihkan snapshot MemoryDB menggunakan operasi API MemoryDB.

CreateCluster

Saat menggunakan `CreateCluster` operasi, pastikan untuk menyertakan parameter `SnapshotName` atau `SnapshotArns` menyemai cluster baru dengan data dari snapshot.

Untuk informasi selengkapnya, lihat berikut ini:

- [Membuat cluster \(MemoryDB API\)](#) di Panduan Pengguna MemoryDB.
- [CreateCluster](#) di Referensi API MemoryDB.

Menyemai cluster baru dengan snapshot yang dibuat secara eksternal

Saat Anda membuat cluster MemoryDB baru, Anda dapat menyemai dengan data dari file snapshot Redis OSS.rdb.

Untuk menyemai cluster MemoryDB baru dari snapshot MemoryDB atau snapshot ElastiCache (Redis OSS), lihat. [Memulihkan dari snapshot](#)

Saat Anda menggunakan file Redis OSS.rdb untuk menyemai cluster MemoryDB baru, Anda dapat melakukan hal berikut:

- Tentukan sejumlah pecahan di cluster baru. Jumlah ini bisa berbeda dari jumlah pecahan di cluster yang digunakan untuk membuat file snapshot.
- Tentukan jenis node yang berbeda untuk cluster baru—lebih besar atau lebih kecil dari yang digunakan dalam cluster yang membuat snapshot. Jika Anda menskalakan ke tipe node yang lebih kecil, pastikan bahwa tipe node baru memiliki memori yang cukup untuk data Anda dan overhead Redis OSS.

Important

- Anda harus memastikan bahwa data snapshot Anda tidak melebihi sumber daya node.

Jika snapshot terlalu besar, cluster yang dihasilkan memiliki status `restore-failed`. Jika hal ini terjadi, Anda harus menghapus klaster tersebut dan memulai dari awal.

Untuk daftar lengkap jenis dan spesifikasi node, lihat [Parameter spesifik tipe node MemoryDB](#).

- Anda dapat mengenkripsi file Redis OSS.rdb dengan enkripsi sisi server Amazon S3 (SSE-S3) saja. Untuk informasi selengkapnya, lihat [Melindungi data menggunakan enkripsi sisi server](#).

Langkah 1: Buat snapshot Redis OSS pada cluster eksternal

Untuk membuat snapshot untuk menyemai cluster MemoryDB Anda

1. Connect ke instans Redis OSS Anda yang ada.

2. Jalankan Redis OSS BGSAVE atau SAVE operasi untuk membuat snapshot. Catat tempat file .rdb Anda berada.

BGSAVE bersifat asinkron dan tidak memblokir klien lain saat melakukan pemrosesan. Untuk informasi lebih lanjut, lihat [BGSAVE di situs](#) web Redis OSS.

SAVE bersifat sinkron dan memblokir proses lainnya hingga selesai. Untuk informasi lebih lanjut, lihat [SIMPAN](#) di situs web Redis OSS.

Untuk informasi tambahan tentang membuat snapshot, lihat [Ketekunan Redis OSS](#) di situs web Redis OSS.

Langkah 2: Buat folder dan bucket Amazon S3

Ketika Anda telah membuat file snapshot, Anda perlu mengunggahnya ke folder dalam ember Amazon S3. Untuk melakukannya, Anda harus memiliki bucket Amazon S3 dan folder dalam bucket tersebut terlebih dahulu. Jika Anda sudah memiliki bucket Amazon S3 dan folder dengan izin yang sesuai, Anda dapat melanjutkan ke [Langkah 3: Unggah snapshot Anda ke Amazon S3](#).


Untuk membuat bucket Amazon S3

1. [Masuk ke AWS Management Console dan buka konsol Amazon S3 di https://console.aws.amazon.com/s3/](https://console.aws.amazon.com/s3/).
2. Ikuti petunjuk untuk membuat bucket Amazon S3 di [Membuat bucket](#) dalam Panduan Pengguna Amazon Simple Storage Service.

Nama bucket Amazon S3 Anda harus sesuai dengan persyaratan DNS. Jika tidak, MemoryDB tidak dapat mengakses file cadangan Anda. Aturan untuk kesesuaian DNS adalah:

- Nama harus minimal 3 dan tidak lebih dari 63 karakter.
- Nama harus serangkaian satu atau beberapa label yang dipisahkan oleh titik (.) dengan setiap label:
 - Dimulai dengan huruf kecil atau angka.
 - Diakhiri dengan huruf kecil atau angka.
 - Hanya berisi huruf kecil, angka, dan tanda hubung.
- Nama tidak dapat diformat sebagai alamat IP (misalnya, 192.0.2.0).

Kami sangat menyarankan Anda membuat bucket Amazon S3 Anda di AWS Wilayah yang sama dengan cluster MemoryDB baru Anda. Pendekatan ini memastikan bahwa kecepatan transfer data tertinggi saat MemoryDB membaca file.rdb Anda dari Amazon S3.

 Note

Untuk menjaga data Anda seaman mungkin, buat izin di bucket Amazon S3 Anda seketat mungkin. Pada saat yang sama, izin masih perlu mengizinkan bucket dan isinya digunakan untuk menyemai cluster MemoryDB baru Anda.

Untuk menambahkan folder ke bucket Amazon S3

1. [Masuk ke AWS Management Console dan buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.](https://console.aws.amazon.com/s3/)
2. Pilih nama bucket untuk mengunggah file .rdb Anda.
3. Pilih Buat folder.
4. Masukkan nama untuk folder baru Anda.
5. Pilih Simpan.

Catat nama dari bucket dan folder.

Langkah 3: Unggah snapshot Anda ke Amazon S3

Sekarang, unggah file .rdb yang Anda buat di [Langkah 1: Buat snapshot Redis OSS pada cluster eksternal](#). Anda mengunggahnya ke bucket Amazon S3 dan folder yang Anda buat di [Langkah 2: Buat folder dan bucket Amazon S3](#). Untuk informasi selengkapnya tentang tugas ini, lihat [Mengunggah objek](#). Di antara langkah 2 dan 3, pilih nama folder yang Anda buat.

Untuk mengunggah file .rdb Anda ke folder Amazon S3

1. [Masuk ke AWS Management Console dan buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.](https://console.aws.amazon.com/s3/)
2. Pilih nama bucket Amazon S3 yang Anda buat di Langkah 2.
3. Pilih nama folder yang Anda buat di Langkah 2.

4. Pilih Unggah.
5. Pilih Tambahkan file.
6. Telusuri untuk mencari file atau beberapa file yang ingin diunggah, lalu pilih file atau beberapa file tersebut. Untuk memilih beberapa file, tahan tombol Ctrl saat memilih setiap nama file.
7. Pilih Buka.
8. Konfirmasikan file atau file yang benar tercantum di halaman Unggah, lalu pilih Unggah.

Catat jalur ke file `.rdb` Anda. Misalnya, jika nama bucket Anda `myBucket` dan jalurnya adalah `myFolder/redis.rdb`, masukkan `myBucket/myFolder/redis.rdb`. Anda memerlukan jalur ini untuk menyemai cluster baru dengan data dalam snapshot ini.

Untuk informasi tambahan, lihat [Aturan penamaan bucket](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Langkah 4: Berikan akses baca MemoryDB ke file.rdb

AWS Wilayah yang diperkenalkan sebelum 20 Maret 2019, diaktifkan secara default. Anda dapat mulai bekerja di AWS Wilayah ini segera. Wilayah yang diperkenalkan setelah 20 Maret 2019 dinonaktifkan secara default. Anda harus mengaktifkan, atau ikut serta, ke Wilayah ini sebelum Anda dapat menggunakannya, seperti yang dijelaskan dalam [Mengelola AWS wilayah](#).

Berikan akses baca MemoryDB ke file.rdb

Untuk memberikan akses baca MemoryDB ke file snapshot

1. [Masuk ke AWS Management Console dan buka konsol Amazon S3 di https://console.aws.amazon.com/s3/](https://console.aws.amazon.com/s3/).
2. Pilih nama bucket S3 yang berisi file `.rdb` Anda.
3. Pilih nama folder yang berisi file `.rdb` Anda.
4. Pilih nama file snapshot `.rdb` Anda. Nama file yang dipilih muncul di atas tab di bagian atas halaman.
5. Pilih tab Izin.
6. Di bagian Izin, pilih Kebijakan bucket, lalu pilih Edit.
7. Perbarui kebijakan untuk memberikan izin yang diperlukan MemoryDB untuk melakukan operasi:
 - Tambahkan [`"Service" : "region-full-name.memorydb-snapshot.amazonaws.com"`] ke Principal.

- Menambahkan izin yang diperlukan berikut untuk mengekspor snapshot ke bucket Amazon S3:
 - "s3:GetObject"
 - "s3:ListBucket"
 - "s3:GetBucketAcl"

Berikut adalah contoh tampilan kebijakan yang sudah diperbarui.

```
{
  "Version": "2012-10-17",
  "Id": "Policy15397346",
  "Statement": [
    {
      "Sid": "Stmt15399483",
      "Effect": "Allow",
      "Principal": {
        "Service": "us-east-1.memorydb-snapshot.amazonaws.com"
      },
      "Action": [
        "s3:GetObject",
        "s3:ListBucket",
        "s3:GetBucketAcl"
      ],
      "Resource": [
        "arn:aws:s3:::example-bucket",
        "arn:aws:s3:::example-bucket/snapshot1.rdb",
        "arn:aws:s3:::example-bucket/snapshot2.rdb"
      ]
    }
  ]
}
```

8. Pilih Simpan.

Langkah 5: Benih cluster MemoryDB dengan data file.rdb

Sekarang Anda siap untuk membuat cluster MemoryDB dan menyemai dengan data dari file.rdb. Untuk membuat cluster, ikuti petunjuk di [Membuat cluster MemoryDB](#).

Metode yang Anda gunakan untuk memberi tahu MemoryDB di mana menemukan snapshot Redis OSS yang Anda unggah ke Amazon S3 bergantung pada metode yang Anda gunakan untuk membuat cluster:

Benih cluster MemoryDB dengan data file.rdb

- Menggunakan konsol MemoryDB

Setelah Anda memilih mesin Redis OSS, perluas bagian pengaturan Advanced Redis OSS dan temukan Impor data ke cluster. Di kotak Lokasi S3 file RDB seed, ketikkan jalur Amazon S3 untuk file tersebut. Jika Anda memiliki beberapa file .rdb, ketikkan jalur untuk setiap file dalam daftar yang dipisahkan koma. Jalur Amazon S3 terlihat seperti *myBucket/myFolder/myBackupFilename*.rdb.

- Menggunakan AWS CLI

Jika Anda menggunakan operasi `create-cluster` atau `create-cluster`, gunakan parameter `--snapshot-arns` untuk menentukan ARN yang memenuhi syarat sepenuhnya untuk setiap file .rdb. Misalnya, `arn:aws:s3:::myBucket/myFolder/myBackupFilename`.rdb. ARN harus menyelesaikan file snapshot yang Anda simpan di Amazon S3.

- Menggunakan MemoryDB API

Jika Anda menggunakan `CreateCluster` atau operasi API `CreateCluster` MemoryDB, gunakan parameter `SnapshotArns` untuk menentukan ARN yang sepenuhnya memenuhi syarat untuk setiap file.rdb. Misalnya, `arn:aws:s3:::myBucket/myFolder/myBackupFilename`.rdb. ARN harus menyelesaikan file snapshot yang Anda simpan di Amazon S3.

Selama proses pembuatan cluster Anda, data dalam snapshot Anda ditulis ke cluster. Anda dapat memantau kemajuan dengan melihat pesan acara MemoryDB. Untuk melakukan ini, lihat konsol MemoryDB dan pilih Acara. Anda juga dapat menggunakan antarmuka baris perintah AWS MemoryDB atau API MemoryDB untuk mendapatkan pesan acara.

Menandai snapshot

Anda dapat menetapkan metadata Anda sendiri untuk setiap snapshot dalam bentuk tag. Tag memungkinkan Anda untuk mengkategorikan snapshot Anda dengan cara yang berbeda, misalnya, berdasarkan tujuan, pemilik, atau lingkungan. Hal ini berguna ketika Anda memiliki banyak sumber daya dengan jenis yang sama—Anda dapat dengan cepat mengidentifikasi sumber daya tertentu berdasarkan tag yang telah Anda tetapkan. Untuk informasi selengkapnya, lihat [Sumber daya yang dapat Anda beri tag](#).

Tag alokasi biaya adalah sarana untuk melacak biaya Anda di beberapa AWS layanan dengan mengelompokkan pengeluaran Anda pada faktur berdasarkan nilai tag. Untuk mempelajari selengkapnya tentang tag alokasi biaya, lihat [Menggunakan tag alokasi biaya](#).

Menggunakan konsol MemoryDB, API AWS CLI, atau MemoryDB Anda dapat menambahkan, membuat daftar, memodifikasi, menghapus, atau menyalin tag alokasi biaya pada snapshot Anda. Untuk informasi selengkapnya, lihat [Memantau biaya dengan tag alokasi biaya](#).

Menghapus snapshot

Snapshot otomatis dihapus secara otomatis ketika batas retensi berakhir. Jika Anda menghapus cluster, semua snapshot otomatisnya juga dihapus.

MemoryDB menyediakan operasi penghapusan API yang memungkinkan Anda menghapus snapshot kapan saja, terlepas dari apakah snapshot dibuat secara otomatis atau manual. Karena snapshot manual tidak memiliki batas retensi, penghapusan manual adalah satu-satunya cara untuk menghapusnya.

Anda dapat menghapus snapshot menggunakan konsol MemoryDB, API AWS CLI, atau MemoryDB.

Menghapus snapshot (Konsol)

Prosedur berikut menghapus snapshot menggunakan konsol MemoryDB.

Untuk menghapus snapshot

1. [Masuk ke AWS Management Console dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/.](https://console.aws.amazon.com/memorydb/)
2. Di panel navigasi kiri, pilih Snapshots.

Layar Snapshots muncul dengan daftar snapshot Anda.

3. Pilih tombol radio di sebelah kiri nama snapshot yang ingin Anda hapus.
4. Pilih Tindakan dan kemudian pilih Hapus.
5. Jika Anda ingin menghapus snapshot ini, masukkan `delete` di kotak teks lalu pilih Hapus. Untuk membatalkan penghapusan, pilih Batal. Status berubah menjadi menghapus.

Menghapus snapshot (CLI AWS)

Gunakan AWS CLI operasi hapus-snapshot dengan parameter berikut untuk menghapus snapshot.

- `--snapshot-name`— Nama snapshot yang akan dihapus.

Kode berikut menghapus snapshot `myBackup`.

```
aws memorydb delete-snapshot --snapshot-name myBackup
```

Untuk informasi selengkapnya, lihat [delete-snapshot](#) dalam Referensi Perintah AWS CLI .

Menghapus snapshot (MemoryDB API)

Gunakan operasi DeleteSnapshot API dengan parameter berikut untuk menghapus snapshot.

- SnapshotName— Nama snapshot yang akan dihapus.

Kode berikut menghapus snapshotmyBackup.

```
https://memory-db.us-east-1.amazonaws.com/
?Action=DeleteSnapshot
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&SnapshotName=myBackup
&Timestamp=20210802T192317Z
&Version=2021-01-01
&X-Amz-Credential=<credential>
```

Untuk informasi lebih lanjut, lihat [DeleteSnapshot](#).

Penskalaan

Jumlah data yang perlu diproses oleh aplikasi Anda jarang bersifat statis. Jumlahnya meningkat dan menurun sejalan dengan bisnis Anda yang bertumbuh atau mengalami fluktuasi permintaan yang normal. Jika Anda mengelola sendiri aplikasi Anda, Anda perlu menyediakan perangkat keras yang cukup untuk puncak permintaan Anda, yang bisa mahal. Dengan menggunakan MemoryDB Anda dapat menskalakan untuk memenuhi permintaan saat ini, hanya membayar untuk apa yang Anda gunakan.

Pertimbangan berikut akan membantu Anda menemukan topik yang tepat untuk tindakan penskalaan yang ingin Anda lakukan.

Penskalaan MemoryDB

Tindakan	MemoryDB
Menskalakan ke luar	Resharding online dan penyeimbangan kembali shard untuk MemoryDB

Tindakan	MemoryDB	
Mengubah jenis simpul	Penskalaan vertikal online dengan mengubah tipe simpul	
Mengubah jumlah pecahan	Penskalaan cluster MemoryDB	

Penskalaan cluster MemoryDB

Ketika permintaan pada cluster Anda berubah, Anda mungkin memutuskan untuk meningkatkan kinerja atau mengurangi biaya dengan mengubah jumlah pecahan di cluster MemoryDB Anda. Sebaiknya gunakan penskalaan horizontal online untuk melakukan tindakan tersebut. Dengan begitu, klaster Anda dapat terus melayani permintaan selama proses penskalaan.

Kondisi berikut dapat menjadi faktor yang membuat Anda memutuskan menskalakan ulang klaster:

- Tekanan memori:

Jika simpul di klaster Anda mengalami tekanan memori, sebaiknya pertimbangkan untuk menskalakan ke luar agar memiliki lebih banyak sumber daya untuk menyimpan data dan melayani permintaan dengan lebih baik.

Anda dapat menentukan apakah node Anda berada di bawah tekanan memori dengan memantau metrik berikut: `FreeableMemory`, `SwapUsage`, dan `BytesUsedForMemoryDB`.

- Hambatan CPU atau jaringan:

Jika masalah latensi/throughput mengganggu klaster, Anda mungkin perlu menskalakan ke luar untuk menyelesaikan masalah.

Anda dapat memantau tingkat latensi dan throughput Anda dengan memantau metrik berikut: `CPUUtilization`,,,, dan `NetworkBytesInNetworkBytesOutCurrConnectionsNewConnections`

- Klaster Anda diskalakan berlebihan:

Permintaan saat ini pada klaster Anda dalam kondisi yang membuat penskalaan ke dalam tidak mengganggu performa dan mengurangi biaya.

Anda dapat memantau penggunaan klaster untuk menentukan apakah Anda dapat menskalakan dengan aman menggunakan metrik berikut: `FreeableMemory`,, `BytesUsedForMemoryDB` `SwapUsage`, `CPUUtilization`,, `NetworkBytesIn`, `NetworkBytesOut` dan `CurrConnectionsNewConnections`

Dampak Performa dari Penskalaan

Ketika Anda menskalakan menggunakan proses offline, klaster akan offline selama sebagian besar proses dan dengan demikian tidak dapat melayani permintaan. Ketika Anda menskalakan menggunakan metode online, karena penskalaan adalah operasi sarat komputasi, ada beberapa

penurunan dalam performa, namun, klaster Anda terus melayani permintaan selama operasi penskalaan. Berapa banyak penurunan yang Anda alami bergantung pada pemanfaatan CPU normal dan data Anda.

Ada dua cara untuk menskalakan cluster MemoryDB Anda; penskalaan horizontal dan vertikal.

- Penskalaan horizontal memungkinkan Anda mengubah jumlah pecahan di cluster dengan menambahkan atau menghapus pecahan. Proses resharding online memungkinkan penskalaan ke dalam/luar sementara klaster terus melayani permintaan masuk.
- Penskalaan Vertikal - Ubah jenis simpul untuk mengubah ukuran klaster. Penskalaan vertikal secara online memungkinkan untuk menaikkan skala/menurunkan skala sementara klaster terus melayani permintaan masuk.

Jika Anda mengurangi ukuran dan kapasitas memori cluster, baik dengan skala atau scaling down, pastikan bahwa konfigurasi baru memiliki memori yang cukup untuk data Anda dan overhead Redis OSS.

Resharding offline dan penyeimbangan kembali shard untuk MemoryDB

Keuntungan utama yang Anda dapatkan dari konfigurasi ulang shard offline adalah Anda dapat melakukan lebih dari sekadar menambah atau menghapus pecahan dari cluster Anda. Saat Anda melakukan reshard offline, selain mengubah jumlah pecahan di cluster Anda, Anda dapat melakukan hal berikut:

- Ubah jenis node cluster Anda.
- Tingkatkan ke versi mesin yang lebih baru.

Note

Resharding offline tidak didukung pada cluster dengan tiering data diaktifkan. Untuk informasi selengkapnya, lihat [Tingkatan data](#).

Kekurangan utama dari konfigurasi ulang serpihan secara offline adalah bahwa klaster Anda offline dimulai dari bagian pemulihan dari proses berlanjut sampai Anda memperbarui titik akhir dalam aplikasi Anda. Durasi offline klaster Anda akan bervariasi tergantung jumlah data dalam klaster Anda.

Untuk mengkonfigurasi ulang pecahan Anda MemoryDB cluster offline

1. Buat snapshot manual dari cluster MemoryDB Anda yang ada. Untuk informasi selengkapnya, lihat [Membuat snapshot manual](#).
2. Buat cluster baru dengan memulihkan dari snapshot. Untuk informasi selengkapnya, lihat [Memulihkan dari snapshot](#).
3. Perbarui titik akhir dalam aplikasi Anda untuk titik akhir klaster baru. Untuk informasi selengkapnya, lihat [Menemukan titik akhir koneksi](#).

Resharding online dan penyeimbangan kembali shard untuk MemoryDB

Dengan menggunakan resharding online dan shard rebalancing dengan MemoryDB, Anda dapat menskalakan MemoryDB Anda secara dinamis tanpa downtime. Pendekatan ini berarti bahwa klaster Anda dapat terus melayani permintaan bahkan saat penskalaan atau penyeimbangan ulang sedang dalam proses.

Anda dapat melakukan tindakan berikut:

- Skalikan - Tingkatkan kapasitas baca dan tulis dengan menambahkan pecahan ke cluster MemoryDB Anda.

Jika Anda menambahkan satu atau lebih pecahan ke cluster Anda, jumlah node di setiap pecahan baru sama dengan jumlah node di pecahan terkecil yang ada.

- Skala - Kurangi kapasitas baca dan tulis, dan dengan demikian biayanya, dengan menghapus pecahan dari cluster MemoryDB Anda.

Saat ini, batasan berikut berlaku untuk resharding online MemoryDB:

- Terdapat batasan dengan slot atau ruang kunci dan item besar:

Jika salah satu kunci dalam serpihan berisi item besar, kunci tersebut tidak dimigrasikan ke serpihan baru ketika penskalaan ke luar atau penyeimbangan ulang dilakukan. Fungsi ini dapat menghasilkan serpihan yang tidak seimbang.

Jika salah satu kunci dalam serpihan berisi item besar (item yang lebih besar dari 256 MB setelah serialisasi), serpihan tersebut tidak dihapus ketika penskalaan ke dalam dilakukan. Fungsi ini dapat mengakibatkan beberapa serpihan tidak dihapus.

- Saat menskalakan, jumlah node di setiap pecahan baru sama dengan jumlah node di pecahan yang ada.

Untuk informasi selengkapnya, lihat [Praktik terbaik: Mengubah ukuran klaster secara online](#).

Anda dapat menskalakan atau menyeimbangkan kembali cluster MemoryDB Anda secara horizontal menggunakan, API AWS Management Console, dan MemoryDB. AWS CLI

Menambahkan serpihan dengan resharding online

Anda dapat menambahkan pecahan ke cluster MemoryDB Anda menggunakan AWS Management Console, AWS CLI, atau MemoryDB API.

Menambahkan serpihan (Konsol)

Anda dapat menggunakan AWS Management Console untuk menambahkan satu atau lebih pecahan ke cluster MemoryDB Anda. Prosedur berikut menjelaskan prosesnya.

1. [Masuk ke AWS Management Console dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/.](https://console.aws.amazon.com/memorydb/)
2. Dari daftar cluster, pilih nama cluster dari mana Anda ingin menambahkan pecahan.
3. Di bawah tab Pecahan dan node, pilih Tambah/Hapus pecahan
4. Di Jumlah pecahan baru, masukkan jumlah pecahan yang Anda inginkan.
5. Pilih Konfirmasi untuk menyimpan perubahan atau Batalkan untuk dibuang.

Menambahkan serpihan (AWS CLI)

Proses berikut menjelaskan cara mengkonfigurasi ulang pecahan di cluster MemoryDB Anda dengan menambahkan pecahan menggunakan file. AWS CLI

Gunakan parameter berikut dengan `update-cluster`.

Parameter

- `--cluster-name` – Wajib. Menentukan cluster (cluster) operasi konfigurasi ulang shard yang akan dilakukan pada.
- `--shard-configuration` – Wajib. Memungkinkan Anda mengatur jumlah pecahan.
 - `ShardCount`— Tetapkan properti ini untuk menentukan jumlah pecahan yang Anda inginkan.

Example

Contoh berikut memodifikasi jumlah pecahan di cluster menjadi 2my-cluster.

Untuk Linux, macOS, atau Unix:

```
aws memorydb update-cluster \  
  --cluster-name my-cluster \  
  --shard-configuration \  
    ShardCount=2
```

Untuk Windows:

```
aws memorydb update-cluster ^  
  --cluster-name my-cluster ^  
  --shard-configuration ^  
    ShardCount=2
```

Ini mengembalikan respon JSON berikut:

```
{  
  "Cluster": {  
    "Name": "my-cluster",  
    "Status": "updating",  
    "NumberOfShards": 2,  
    "AvailabilityMode": "MultiAZ",  
    "ClusterEndpoint": {  
      "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-east-1.amazonaws.com",  
      "Port": 6379  
    },  
    "NodeType": "db.r6g.large",  
    "EngineVersion": "6.2",  
    "EnginePatchVersion": "6.2.6",  
    "ParameterGroupName": "default.memorydb-redis6",  
    "ParameterGroupStatus": "in-sync",  
    "SubnetGroupName": "my-sg",  
    "TLSEnabled": true,  
    "ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",  
    "SnapshotRetentionLimit": 0,  
    "MaintenanceWindow": "wed:03:00-wed:04:00",  
    "SnapshotWindow": "04:30-05:30",  
    "DataTiering": "false",  
    "AutoMinorVersionUpgrade": true  
  }  
}
```

```
}  
}
```

Untuk melihat detail klaster yang diperbarui setelah statusnya berubah dari pembaruan menjadi tersedia, gunakan perintah berikut:

Untuk Linux, macOS, atau Unix:

```
aws memorydb describe-clusters \  
  --cluster-name my-cluster  
  --show-shard-details
```

Untuk Windows:

```
aws memorydb describe-clusters ^  
  --cluster-name my-cluster  
  --show-shard-details
```

Ini akan mengembalikan respons JSON berikut:

```
{  
  "Clusters": [  
    {  
      "Name": "my-cluster",  
      "Status": "available",  
      "NumberOfShards": 2,  
      "Shards": [  
        {  
          "Name": "0001",  
          "Status": "available",  
          "Slots": "0-8191",  
          "Nodes": [  
            {  
              "Name": "my-cluster-0001-001",  
              "Status": "available",  
              "AvailabilityZone": "us-east-1a",  
              "CreateTime": "2021-08-21T20:22:12.405000-07:00",  
              "Endpoint": {  
                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-  
east-1.amazonaws.com",  
                "Port": 6379
```

```

    }
  },
  {
    "Name": "my-cluster-0001-002",
    "Status": "available",
    "AvailabilityZone": "us-east-1b",
    "CreateTime": "2021-08-21T20:22:12.405000-07:00",
    "Endpoint": {
      "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
      "Port": 6379
    }
  }
],
"NumberOfNodes": 2
},
{
  "Name": "0002",
  "Status": "available",
  "Slots": "8192-16383",
  "Nodes": [
    {
      "Name": "my-cluster-0002-001",
      "Status": "available",
      "AvailabilityZone": "us-east-1b",
      "CreateTime": "2021-08-22T14:26:18.693000-07:00",
      "Endpoint": {
        "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
        "Port": 6379
      }
    },
    {
      "Name": "my-cluster-0002-002",
      "Status": "available",
      "AvailabilityZone": "us-east-1a",
      "CreateTime": "2021-08-22T14:26:18.765000-07:00",
      "Endpoint": {
        "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
        "Port": 6379
      }
    }
  ]
},
],

```



```

        "NumberOfNodes": 2
      }
    ],
    "ClusterEndpoint": {
      "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-east-1.amazonaws.com",
      "Port": 6379
    },
    "NodeType": "db.r6g.large",
    "EngineVersion": "6.2",
    "EnginePatchVersion": "6.2.6",
    "ParameterGroupName": "default.memorydb-redis6",
    "ParameterGroupStatus": "in-sync",
    "SubnetGroupName": "my-sg",
    "TLSEnabled": true,
    "ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",
    "SnapshotRetentionLimit": 0,
    "MaintenanceWindow": "wed:03:00-wed:04:00",
    "SnapshotWindow": "04:30-05:30",
    "ACLName": "my-acl",
    "DataTiering": "false",
    "AutoMinorVersionUpgrade": true
  }
]
}

```

Untuk informasi selengkapnya, lihat [update-cluster](#) di Command Reference. AWS CLI

Menambahkan pecahan (MemoryDB API)

Anda dapat menggunakan API MemoryDB untuk mengkonfigurasi ulang pecahan di cluster MemoryDB Anda secara online dengan menggunakan operasi. UpdateCluster

Gunakan parameter berikut dengan UpdateCluster.

Parameter

- **ClusterName** – Wajib. Menentukan cluster mana operasi konfigurasi ulang shard akan dilakukan pada.
- **ShardConfiguration** – Wajib. Memungkinkan Anda mengatur jumlah pecahan.
 - **ShardCount**— Tetapkan properti ini untuk menentukan jumlah pecahan yang Anda inginkan.


Untuk informasi lebih lanjut, lihat [UpdateCluster](#).

Menghapus serpihan dengan resharding online

Anda dapat menghapus pecahan dari cluster MemoryDB Anda menggunakan AWS Management Console, AWS CLI, atau MemoryDB API.

Menghapus serpihan (Konsol)

Proses berikut menjelaskan cara mengkonfigurasi ulang pecahan di cluster MemoryDB Anda dengan menghapus pecahan menggunakan file. AWS Management Console

 Important

Sebelum menghapus pecahan dari cluster Anda, MemoryDB memastikan bahwa semua data Anda akan muat di pecahan yang tersisa. Jika data akan cocok, pecahan dihapus dari cluster seperti yang diminta. Jika data tidak sesuai dengan pecahan yang tersisa, proses dihentikan dan cluster dibiarkan dengan konfigurasi pecahan yang sama seperti sebelum permintaan dibuat.

Anda dapat menggunakan AWS Management Console untuk menghapus satu atau lebih pecahan dari cluster MemoryDB Anda. Anda tidak dapat menghapus semua pecahan dalam sebuah cluster. Sebagai gantinya, Anda harus menghapus cluster. Untuk informasi selengkapnya, lihat [Langkah 4: Menghapus cluster](#). Prosedur berikut menjelaskan proses untuk menghapus satu atau lebih pecahan.

1. [Masuk ke AWS Management Console dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/](https://console.aws.amazon.com/memorydb/).
2. Dari daftar cluster, pilih nama cluster dari mana Anda ingin menghapus pecahan.
3. Di bawah tab Pecahan dan node, pilih Tambah/Hapus pecahan
4. Di Jumlah pecahan baru, masukkan jumlah pecahan yang Anda inginkan (dengan minimal 1).
5. Pilih Konfirmasi untuk menyimpan perubahan atau Batalkan untuk dibuang.

Menghapus serpihan (AWS CLI)

Proses berikut menjelaskan cara mengkonfigurasi ulang pecahan di cluster MemoryDB Anda dengan menghapus pecahan menggunakan file. AWS CLI

⚠ Important

Sebelum menghapus pecahan dari cluster Anda, MemoryDB memastikan bahwa semua data Anda akan muat di pecahan yang tersisa. Jika data cocok, pecahan dihapus dari cluster seperti yang diminta dan ruang kuncinya dipetakan ke pecahan yang tersisa. Jika data tidak sesuai dengan pecahan yang tersisa, proses dihentikan dan cluster dibiarkan dengan konfigurasi pecahan yang sama seperti sebelum permintaan dibuat.

Anda dapat menggunakan AWS CLI untuk menghapus satu atau lebih pecahan dari cluster MemoryDB Anda. Anda tidak dapat menghapus semua pecahan dalam sebuah cluster. Sebagai gantinya, Anda harus menghapus cluster. Untuk informasi selengkapnya, lihat [Langkah 4: Menghapus cluster](#).

Gunakan parameter berikut dengan `update-cluster`.

Parameter

- `--cluster-name` – Wajib. Menentukan cluster (cluster) operasi konfigurasi ulang shard yang akan dilakukan pada.
- `--shard-configuration` – Wajib. Memungkinkan Anda mengatur jumlah pecahan menggunakan `ShardCount` properti:

`ShardCount`— Tetapkan properti ini untuk menentukan jumlah pecahan yang Anda inginkan.

Example

Contoh berikut memodifikasi jumlah pecahan di cluster menjadi `2my-cluster`.

Untuk Linux, macOS, atau Unix:

```
aws memorydb update-cluster \  
  --cluster-name my-cluster \  
  --shard-configuration \  
    ShardCount=2
```

Untuk Windows:

```
aws memorydb update-cluster ^
```

```
--cluster-name my-cluster ^
--shard-configuration ^
    ShardCount=2
```

Ini mengembalikan respon JSON berikut:

```
{
  "Cluster": {
    "Name": "my-cluster",
    "Status": "updating",
    "NumberOfShards": 2,
    "AvailabilityMode": "MultiAZ",
    "ClusterEndpoint": {
      "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-east-1.amazonaws.com",
      "Port": 6379
    },
    "NodeType": "db.r6g.large",
    "EngineVersion": "6.2",
    "EnginePatchVersion": "6.2.6",
    "ParameterGroupName": "default.memorydb-redis6",
    "ParameterGroupStatus": "in-sync",
    "SubnetGroupName": "my-sg",
    "TLSEnabled": true,
    "ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",
    "SnapshotRetentionLimit": 0,
    "MaintenanceWindow": "wed:03:00-wed:04:00",
    "SnapshotWindow": "04:30-05:30",
    "DataTiering": "false",
    "AutoMinorVersionUpgrade": true
  }
}
```

Untuk melihat detail klaster yang diperbarui setelah statusnya berubah dari pembaruan menjadi tersedia, gunakan perintah berikut:

Untuk Linux, macOS, atau Unix:

```
aws memorydb describe-clusters \
  --cluster-name my-cluster
  --show-shard-details
```

Untuk Windows:

```
aws memorydb describe-clusters ^
  --cluster-name my-cluster
  --show-shard-details
```

Ini akan mengembalikan respons JSON berikut:

```
{
  "Clusters": [
    {
      "Name": "my-cluster",
      "Status": "available",
      "NumberOfShards": 2,
      "Shards": [
        {
          "Name": "0001",
          "Status": "available",
          "Slots": "0-8191",
          "Nodes": [
            {
              "Name": "my-cluster-0001-001",
              "Status": "available",
              "AvailabilityZone": "us-east-1a",
              "CreateTime": "2021-08-21T20:22:12.405000-07:00",
              "Endpoint": {
                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                "Port": 6379
              }
            },
            {
              "Name": "my-cluster-0001-002",
              "Status": "available",
              "AvailabilityZone": "us-east-1b",
              "CreateTime": "2021-08-21T20:22:12.405000-07:00",
              "Endpoint": {
                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                "Port": 6379
              }
            }
          ],
          "NumberOfNodes": 2
        }
      ]
    }
  ]
}
```

```

    },
    {
      "Name": "0002",
      "Status": "available",
      "Slots": "8192-16383",
      "Nodes": [
        {
          "Name": "my-cluster-0002-001",
          "Status": "available",
          "AvailabilityZone": "us-east-1b",
          "CreateTime": "2021-08-22T14:26:18.693000-07:00",
          "Endpoint": {
            "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
            "Port": 6379
          }
        },
        {
          "Name": "my-cluster-0002-002",
          "Status": "available",
          "AvailabilityZone": "us-east-1a",
          "CreateTime": "2021-08-22T14:26:18.765000-07:00",
          "Endpoint": {
            "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
            "Port": 6379
          }
        }
      ],
      "NumberOfNodes": 2
    }
  ],
  "ClusterEndpoint": {
    "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
    "Port": 6379
  },
  "NodeType": "db.r6g.large",
  "EngineVersion": "6.2",
  "EnginePatchVersion": "6.2.6",
  "ParameterGroupName": "default.memorydb-redis6",
  "ParameterGroupStatus": "in-sync",
  "SubnetGroupName": "my-sg",
  "TLSEnabled": true,

```

```

    "ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",
    "SnapshotRetentionLimit": 0,
    "MaintenanceWindow": "wed:03:00-wed:04:00",
    "SnapshotWindow": "04:30-05:30",
    "ACLName": "my-acl",
    "DataTiering": "false",
    "AutoMinorVersionUpgrade": true
  }
]
}

```

Untuk informasi selengkapnya, lihat [update-cluster](#) di Command Reference. AWS CLI

Menghapus pecahan (MemoryDB API)

Anda dapat menggunakan API MemoryDB untuk mengkonfigurasi ulang pecahan di cluster MemoryDB Anda secara online dengan menggunakan operasi. `UpdateCluster`

Proses berikut menjelaskan cara mengkonfigurasi ulang pecahan di cluster MemoryDB Anda dengan menghapus pecahan menggunakan API MemoryDB.

Important

Sebelum menghapus pecahan dari cluster Anda, MemoryDB memastikan bahwa semua data Anda akan muat di pecahan yang tersisa. Jika data cocok, pecahan dihapus dari cluster seperti yang diminta dan ruang kuncinya dipetakan ke pecahan yang tersisa. Jika data tidak sesuai dengan pecahan yang tersisa, proses dihentikan dan cluster dibiarkan dengan konfigurasi pecahan yang sama seperti sebelum permintaan dibuat.

Anda dapat menggunakan API MemoryDB untuk menghapus satu atau lebih pecahan dari cluster MemoryDB Anda. Anda tidak dapat menghapus semua pecahan dalam sebuah cluster. Sebagai gantinya, Anda harus menghapus cluster. Untuk informasi selengkapnya, lihat [Langkah 4: Menghapus cluster](#).

Gunakan parameter berikut dengan `UpdateCluster`.

Parameter

- `ClusterName` – Wajib. Menentukan cluster (cluster) operasi konfigurasi ulang shard yang akan dilakukan pada.

- `ShardConfiguration` – Wajib. Memungkinkan Anda mengatur jumlah pecahan menggunakan `ShardCount` properti:

`ShardCount`— Tetapkan properti ini untuk menentukan jumlah pecahan yang Anda inginkan.

Penskalaan vertikal online dengan mengubah tipe simpul

Dengan menggunakan penskalaan vertikal online dengan MemoryDB, Anda dapat menskalakan cluster Anda secara dinamis dengan waktu henti minimal. Ini memungkinkan klaster Anda untuk melayani permintaan bahkan saat penskalaan.

Note

Penskalaan tidak didukung antara klaster tingkatan data (misalnya, klaster yang menggunakan jenis simpul `r6gd`) dan klaster yang tidak menggunakan tingkatan data (misalnya, klaster yang menggunakan jenis simpul `r6g`). Untuk informasi selengkapnya, lihat [Tingkatan data](#).

Anda dapat melakukan tindakan berikut:

- **Scale up** — Tingkatkan kapasitas baca dan tulis dengan menyesuaikan tipe node cluster MemoryDB Anda untuk menggunakan tipe node yang lebih besar.

MemoryDB secara dinamis mengubah ukuran cluster Anda sambil tetap online dan melayani permintaan.

- **Menurunkan skala** – Mengurangi kapasitas baca dan tulis dengan menyesuaikan jenis simpul agar menggunakan simpul yang lebih kecil. Sekali lagi, MemoryDB secara dinamis mengubah ukuran cluster Anda sambil tetap online dan melayani permintaan. Dalam hal ini, Anda mengurangi biaya dengan menurunkan ukuran simpul.

Note

Proses kenaikan dan penurunan skala bergantung pada pembuatan klaster dengan jenis simpul yang baru dipilih dan sinkronisasi simpul baru dengan yang sebelumnya. Untuk memastikan agar alur kenaikan dan penurunan skala berjalan lancar, lakukan hal berikut:

- Sementara proses penskalaan vertikal dirancang untuk tetap sepenuhnya online, hal ini tidak bergantung pada sinkronisasi data antara simpul lama dan simpul baru. Sebaiknya mulai kenaikan/penurunan skala selama jam ketika lalu lintas data diperkirakan berada pada tingkat minimum.
- Uji perilaku aplikasi Anda selama penskalaan ke dalam di lingkungan persiapan, jika memungkinkan.

Penskalaan ke atas secara online

Topik

- [Meningkatkan kluster MemoryDB \(Konsol\)](#)
- [Meningkatkan kluster MemoryDB \(CLI\)AWS](#)
- [Meningkatkan kluster MemoryDB \(MemoryDB API\)](#)

Meningkatkan kluster MemoryDB (Konsol)

Prosedur berikut menjelaskan cara meningkatkan cluster MemoryDB menggunakan file. AWS Management Console Selama proses ini, cluster MemoryDB Anda akan terus melayani permintaan dengan downtime minimal.

Untuk meningkatkan skala cluster (konsol)

1. [Masuk ke AWS Management Console dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/.](https://console.aws.amazon.com/memorydb/)
2. Pada daftar klaster, pilih klaster tersebut.
3. Pilih Tindakan, lalu pilih Ubah.
4. Dalam dialog Modify Cluster:
 - Pilih jenis simpul sebagai tujuan penskalaan dari daftar Jenis simpul. Untuk menaikkan skala, pilih jenis simpul yang lebih besar dari simpul saat ini.
5. Pilih Simpan perubahan.

Status cluster berubah menjadi modifikasi. Ketika status berubah ke tersedia, perubahan selesai dan Anda dapat mulai menggunakan klaster baru tersebut.

Meningkatkan kluster MemoryDB (CLI)AWS

Prosedur berikut menjelaskan cara meningkatkan cluster MemoryDB menggunakan file. AWS CLI Selama proses ini, cluster MemoryDB Anda akan terus melayani permintaan dengan downtime minimal.

Untuk meningkatkan cluster MemoryDB (CLI AWS)

1. Tentukan jenis node yang dapat Anda tingkatkan dengan menjalankan AWS CLI `list-allowed-node-type-updates` perintah dengan parameter berikut.

Untuk Linux, macOS, atau Unix:

```
aws memorydb list-allowed-node-type-updates \  
  --cluster-name my-cluster-name
```

Untuk Windows:

```
aws memorydb list-allowed-node-type-updates ^  
  --cluster-name my-cluster-name
```

Output dari perintah di atas terlihat seperti berikut (format JSON).

```
{  
  "ScaleUpNodeTypes": [  
    "db.r6g.2xlarge",  
    "db.r6g.large"  
  ],  
  "ScaleDownNodeTypes": [  
    "db.r6g.large"  
  ],  
}
```

Untuk informasi selengkapnya, lihat [list-allowed-node-type-update](#) di AWS CLI Referensi.

2. Ubah cluster Anda untuk meningkatkan ke tipe node baru yang lebih besar, menggunakan AWS CLI `update-cluster` perintah dan parameter berikut.
 - `--cluster-name`— Nama cluster yang Anda tingkatkan.

- `--node-type`— Jenis node baru yang ingin Anda skalakan cluster. Nilai ini harus berupa salah satu dari jenis simpul yang dihasilkan oleh perintah `list-allowed-node-type-updates` di langkah 1.

Untuk Linux, macOS, atau Unix:

```
aws memorydb update-cluster \  
  --cluster-name my-cluster \  
  --node-type db.r6g.2xlarge
```

Untuk Windows:

```
aws memorydb update-cluster ^  
  --cluster-name my-cluster ^  
  --node-type db.r6g.2xlarge ^
```

Untuk informasi selengkapnya, lihat [update-cluster](#).

Meningkatkan klaster MemoryDB (MemoryDB API)

Proses berikut menskalakan klaster Anda dari tipe node saat ini ke tipe node baru yang lebih besar menggunakan API MemoryDB. Selama proses ini, MemoryDB memperbarui entri DNS sehingga mereka menunjuk ke node baru. Anda dapat menskalakan kluster yang diaktifkan failover otomatis sementara klaster terus online dan melayani permintaan yang masuk.

Jumlah waktu yang diperlukan untuk menskalakan hingga tipe node yang lebih besar bervariasi, tergantung pada jenis node Anda dan jumlah data di cluster Anda saat ini.

Untuk meningkatkan cluster MemoryDB (MemoryDB API)

1. Tentukan tipe node mana yang dapat Anda tingkatkan untuk menggunakan `ListAllowedNodeTypeUpdates` aksi API MemoryDB dengan parameter berikut.
 - `ClusterName`— nama cluster. Gunakan parameter ini untuk mendeskripsikan cluster tertentu daripada semua cluster.

```
https://memory-db.us-east-1.amazonaws.com/
```

```
?Action=ListAllowedNodeTypeUpdates
&ClusterName=MyCluster
&Version=2021-01-01
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20210802T192317Z
&X-Amz-Credential=<credential>
```

Untuk informasi selengkapnya, lihat [ListAllowedNodeTypeUpdates](#) di Referensi API MemoryDB.

2. Skala cluster Anda saat ini hingga tipe node baru menggunakan aksi API `UpdateCluster` MemoryDB dan dengan parameter berikut.
 - `ClusterName`— nama cluster.
 - `NodeType`— tipe node baru yang lebih besar dari cluster di cluster ini. Nilai ini harus berupa salah satu jenis instans yang ditampilkan oleh tindakan `ListAllowedNodeTypeUpdates` di langkah 1.

```
https://memory-db.us-east-1.amazonaws.com/
?Action=UpdateCluster
&NodeType=db.r6g.2xlarge
&ClusterName=myCluster
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20210801T220302Z
&Version=2021-01-01
&X-Amz-Algorithm=Amazon4-HMAC-SHA256
&X-Amz-Date=20210801T220302Z
&X-Amz-SignedHeaders=Host
&X-Amz-Expires=20210801T220302Z
&X-Amz-Credential=<credential>
&X-Amz-Signature=<signature>
```

Untuk informasi lebih lanjut, lihat [UpdateCluster](#).

Penskalaan ke bawah secara online

Topik

- [Menskalakan cluster MemoryDB \(Konsol\)](#)

- [Menskalakan cluster MemoryDB \(CLI\)AWS](#)
- [Menskalakan cluster MemoryDB \(MemoryDB API\)](#)

Menskalakan cluster MemoryDB (Konsol)

Prosedur berikut menjelaskan cara menurunkan cluster MemoryDB menggunakan AWS Management Console. Selama proses ini, cluster MemoryDB Anda akan terus melayani permintaan dengan downtime minimal.

Untuk menurunkan kluster MemoryDB (konsol)

1. [Masuk ke AWS Management Console dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/.](https://console.aws.amazon.com/memorydb/)
2. Dari daftar kluster, pilih kluster pilihan Anda.
3. Pilih Tindakan, lalu pilih Ubah.
4. Dalam dialog Modify Cluster:
 - Pilih jenis simpul sebagai tujuan penskalaan dari daftar Jenis simpul. Untuk menurunkan skala, pilih jenis simpul yang lebih kecil dari simpul saat ini. Perhatikan bahwa tidak semua jenis simpul tersedia sebagai pilihan penurunan skala.
5. Pilih Simpan perubahan.

Status cluster berubah menjadi modifikasi. Ketika status berubah ke tersedia, perubahan selesai dan Anda dapat mulai menggunakan kluster baru tersebut.

Menskalakan cluster MemoryDB (CLI)AWS

Prosedur berikut menjelaskan cara menurunkan cluster MemoryDB menggunakan AWS CLI. Selama proses ini, cluster MemoryDB Anda akan terus melayani permintaan dengan downtime minimal.

Untuk menurunkan kluster MemoryDB (CLI AWS)

1. Tentukan jenis node yang dapat Anda turunkan dengan menjalankan AWS CLI `list-allowed-node-type-updates` perintah dengan parameter berikut.

Untuk Linux, macOS, atau Unix:

```
aws memorydb list-allowed-node-type-updates \
```

```
--cluster-name my-cluster-name
```

Untuk Windows:

```
aws memorydb list-allowed-node-type-updates ^  
  --cluster-name my-cluster-name
```

Output dari perintah di atas terlihat seperti berikut (format JSON).

```
{  
  "ScaleUpNodeTypes": [  
    "db.r6g.2xlarge",  
    "db.r6g.large"  
  ],  
  "ScaleDownNodeTypes": [  
    "db.r6g.large"  
  ],  
}
```

Untuk informasi selengkapnya, lihat [list-allowed-node-type-update](#).

2. Ubah cluster Anda untuk menurunkan ke tipe node baru yang lebih kecil, menggunakan `update-cluster` perintah dan parameter berikut.

- `--cluster-name`— Nama cluster yang Anda perkecil.
- `--node-type`— Jenis node baru yang ingin Anda skalakan cluster. Nilai ini harus berupa salah satu dari jenis simpul yang dihasilkan oleh perintah `list-allowed-node-type-updates` di langkah 1.

Untuk Linux, macOS, atau Unix:

```
aws memorydb update-cluster \  
  --cluster-name my-cluster \  
  --node-type db.r6g.large
```

Untuk Windows:

```
aws memorydb update-cluster ^  
  --cluster-name my-cluster ^
```

```
--node-type db.r6g.large
```

Untuk informasi selengkapnya, lihat [update-cluster](#).

Menskalakan cluster MemoryDB (MemoryDB API)

Proses berikut menskalakan klaster Anda dari tipe node saat ini ke tipe node baru yang lebih kecil menggunakan API MemoryDB. Selama proses ini, cluster MemoryDB Anda akan terus melayani permintaan dengan downtime minimal.

Jumlah waktu yang diperlukan untuk menurunkan ke tipe node yang lebih kecil bervariasi, tergantung pada jenis node Anda dan jumlah data di cluster Anda saat ini.

Penskalaan turun (MemoryDB API)

1. Tentukan tipe node mana yang dapat Anda turunkan untuk menggunakan [ListAllowedNodeTypeUpdates](#) API dengan parameter berikut:

- `ClusterName`— nama cluster. Gunakan parameter ini untuk mendeskripsikan cluster tertentu daripada semua cluster.

```
https://memory-db.us-east-1.amazonaws.com/  
?Action=ListAllowedNodeTypeUpdates  
&ClusterName=MyCluster  
&Version=2021-01-01  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20210802T192317Z  
&X-Amz-Credential=<credential>
```

2. Skalakan klaster Anda saat ini ke tipe node baru menggunakan [UpdateCluster](#) API dengan parameter berikut.

- `ClusterName`— nama cluster.
- `NodeType`— tipe node baru yang lebih kecil dari cluster di cluster ini. Nilai ini harus berupa salah satu jenis instans yang ditampilkan oleh tindakan `ListAllowedNodeTypeUpdates` di langkah 1.

```
https://memory-db.us-east-1.amazonaws.com/  
?Action=UpdateCluster  
&NodeType=db.r6g.2xlarge  
&ClusterName=myReplGroup  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20210801T220302Z  
&Version=2021-01-01  
&X-Amz-Algorithm=Amazon4-HMAC-SHA256  
&X-Amz-Date=20210801T220302Z  
&X-Amz-SignedHeaders=Host  
&X-Amz-Expires=20210801T220302Z  
&X-Amz-Credential=<credential>  
&X-Amz-Signature=<signature>
```

Mengonfigurasi parameter mesin menggunakan grup parameter

MemoryDB menggunakan parameter untuk mengontrol properti runtime node dan cluster Anda. Secara umum, versi mesin yang lebih baru mencakup parameter tambahan untuk mendukung fungsionalitas yang lebih baru. Untuk mengetahui tabel parameter, lihat [Parameter spesifik Redis OSS](#).

Seperti yang Anda harapkan, beberapa nilai parameter, seperti `maxmemory`, ditentukan oleh mesin dan jenis simpul. Untuk mengetahui tabel nilai parameter ini berdasarkan jenis simpul, lihat [Parameter spesifik tipe node MemoryDB](#).

Topik

- [Manajemen parameter](#)
- [Tingkatan grup parameter](#)
- [Membuat grup parameter](#)
- [Membuat daftar grup parameter berdasarkan nama](#)
- [Membuat daftar nilai grup parameter](#)
- [Mengubah grup parameter](#)
- [Menghapus grup parameter](#)
- [Parameter spesifik Redis OSS](#)

Manajemen parameter

Parameter dikelompokkan bersama ke grup parameter bernama untuk manajemen parameter yang lebih mudah. Grup parameter merepresentasikan kombinasi nilai tertentu untuk parameter yang diteruskan ke perangkat lunak mesin selama pengaktifan. Nilai ini menentukan perilaku berbagai proses mesin di setiap simpul pada saat runtime. Nilai parameter pada grup parameter tertentu berlaku untuk semua simpul yang terkait dengan grup ini, terlepas dari klaster yang memiliki simpul tersebut.

Untuk menyesuaikan performa klaster, Anda dapat mengubah beberapa nilai parameter atau mengubah grup parameter klaster.

- Anda tidak dapat mengubah atau menghapus grup parameter default. Jika membutuhkan nilai parameter kustom, Anda harus membuat grup parameter kustom.
- Keluarga grup parameter dan klaster tempat Anda menetapkan grup parameter tersebut harus kompatibel. Misalnya, jika klaster Anda menjalankan Redis OSS versi 6, Anda hanya dapat menggunakan grup parameter, default atau kustom, dari keluarga `memorydb_redis6`.
- Saat Anda mengubah parameter klaster, perubahan tersebut akan segera diterapkan pada klaster tersebut. Hal ini berlaku terlepas dari apakah Anda mengubah grup parameter klaster itu sendiri atau nilai parameter dalam grup parameter klaster.

Tingkatan grup parameter

Tingkat kelompok parameter MemoryDB

Default Global

Grup parameter root tingkat atas untuk semua pelanggan MemoryDB di wilayah tersebut.

Grup parameter default global:

- Dicapangkan untuk MemoryDB dan tidak tersedia untuk pelanggan.

Default Pelanggan

Salinan grup parameter Global Default yang dibuat untuk digunakan pelanggan.

Grup parameter Default Pelanggan:

- Dibuat dan dimiliki oleh MemoryDB.
- Tersedia untuk pelanggan untuk digunakan sebagai grup parameter untuk setiap cluster yang menjalankan versi mesin yang didukung oleh grup parameter ini.
- Tidak dapat diedit oleh pelanggan.

Milik Pelanggan

Salinan grup parameter Default Pelanggan. Grup parameter Customer Owned dibuat setiap kali pelanggan membuat grup parameter.

Kelompok parameter yang dimiliki pelanggan:

- Dibuat dan dimiliki oleh pelanggan.
- Dapat ditetapkan untuk salah satu klaster pelanggan yang kompatibel.
- Dapat dimodifikasi oleh pelanggan untuk membuat grup parameter kustom.

Tidak semua nilai parameter dapat diubah. Untuk informasi selengkapnya, lihat [Parameter spesifik Redis OSS](#).

Membuat grup parameter

Anda perlu membuat grup parameter baru jika ada satu atau beberapa nilai parameter yang ingin Anda ubah dari nilai default. Anda dapat membuat grup parameter menggunakan konsol MemoryDB, API AWS CLI, atau MemoryDB.

Membuat grup parameter (Konsol)

Prosedur berikut menunjukkan cara membuat grup parameter menggunakan konsol MemoryDB.

Untuk membuat grup parameter menggunakan konsol MemoryDB

1. [Masuk ke AWS Management Console dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/.](https://console.aws.amazon.com/memorydb/)
2. Untuk melihat daftar semua grup parameter yang tersedia, di panel navigasi sebelah kiri pilih Grup Parameter.
3. Untuk membuat grup parameter, pilih Buat grup parameter.

Halaman grup Create parameter muncul.

4. Di kotak Nama, ketik nama unik untuk grup parameter ini.

Saat membuat klaster atau mengubah grup parameter klaster, Anda akan memilih grup parameter berdasarkan namanya. Oleh karena itu, kami merekomendasikan agar namanya informatif dan dapat mengidentifikasi keluarga grup parameter.

Batasan penamaan grup parameter adalah sebagai berikut:

- Harus diawali dengan huruf ASCII.
 - Hanya dapat berisi huruf ASCII, angka, dan tanda hubung.
 - Harus memiliki panjang 1-255 karakter.
 - Tidak boleh berisi dua tanda hubung berurutan.
 - Tidak boleh diakhiri dengan sebuah tanda hubung.
5. Di kotak Deskripsi, ketik deskripsi untuk grup parameter.
 6. Di kotak kompatibilitas versi Redis OSS, pilih versi mesin yang sesuai dengan grup parameter ini.
 7. Di Tag, secara opsional tambahkan tag untuk mencari dan memfilter grup parameter Anda atau melacak AWS biaya Anda.

8. Untuk membuat grup parameter, pilih **Buat**.

Untuk mengakhiri proses tanpa membuat grup parameter, pilih **Batal**.

9. Ketika grup parameter dibuat, grup parameter ini akan memiliki nilai default keluarga. Untuk mengubah nilai default, Anda harus mengubah grup parameter. Untuk informasi selengkapnya, lihat [Mengubah grup parameter](#).

Membuat grup parameter (AWS CLI)

Untuk membuat grup parameter menggunakan AWS CLI, gunakan perintah `create-parameter-group` dengan parameter ini.

- `--parameter-group-name` – Nama grup parameter.

Batasan penamaan grup parameter adalah sebagai berikut:

- Harus diawali dengan huruf ASCII.
- Hanya dapat berisi huruf ASCII, angka, dan tanda hubung.
- Harus memiliki panjang 1-255 karakter.
- Tidak boleh berisi dua tanda hubung berurutan.
- Tidak boleh diakhiri dengan tanda hubung.
- `--family` – Keluarga mesin dan versi untuk grup parameter.
- `--description` – Deskripsi yang diberikan pengguna untuk grup parameter.

Example

Contoh berikut membuat grup parameter bernama `MyRedis6x` menggunakan keluarga `memorydb_redis6` sebagai template.

Untuk Linux, macOS, atau Unix:

```
aws memorydb create-parameter-group \  
  --parameter-group-name myRedis6x \  
  --family memorydb_redis6 \  
  --description "My first parameter group"
```

Untuk Windows:

```
aws memorydb create-parameter-group ^
  --parameter-group-name myRedis6x ^
  --family memorydb_redis6 ^
  --description "My first parameter group"
```

Output dari perintah ini akan terlihat seperti ini.

```
{
  "ParameterGroup": {
    "Name": "myRedis6x",
    "Family": "memorydb_redis6",
    "Description": "My first parameter group",
    "ARN": "arn:aws:memorydb:us-east-1:012345678912:parametergroup/myredis6x"
  }
}
```

Ketika grup parameter dibuat, grup parameter ini akan memiliki nilai default keluarga. Untuk mengubah nilai default, Anda harus mengubah grup parameter. Untuk informasi selengkapnya, lihat [Mengubah grup parameter](#).

Untuk informasi selengkapnya, lihat [create-parameter-group](#).

Membuat grup parameter (MemoryDB API)

Untuk membuat grup parameter menggunakan API MemoryDB, gunakan `CreateParameterGroup` tindakan dengan parameter ini.

- `ParameterGroupName` – Nama grup parameter.

Batasan penamaan grup parameter adalah sebagai berikut:

- Harus diawali dengan huruf ASCII.
- Hanya dapat berisi huruf ASCII, angka, dan tanda hubung.
- Harus memiliki panjang 1-255 karakter.
- Tidak boleh berisi dua tanda hubung berurutan.
- Tidak boleh diakhiri dengan tanda hubung.
- `Family` – Keluarga mesin dan versi untuk grup parameter. Misalnya, `memorydb_redis6`.
- `Description` – Deskripsi yang diberikan pengguna untuk grup parameter.

Example

Contoh berikut membuat grup parameter bernama MyRedis6x menggunakan keluarga memorydb_redis6 sebagai template.

```
https://memory-db.us-east-1.amazonaws.com/  
?Action=CreateParameterGroup  
&Family=memorydb_redis6  
&ParameterGroupName=myRedis6x  
&Description=My%20first%20parameter%20group  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20210802T192317Z  
&Version=2021-01-01  
&X-Amz-Credential=<credential>
```

Respons dari tindakan ini akan terlihat seperti ini.

```
<CreateParameterGroupResponse xmlns="http://memory-db.us-east-1.amazonaws.com/  
doc/2021-01-01/">  
  <CreateParameterGroupResult>  
    <ParameterGroup>  
      <Name>myRedis6x</Name>  
      <Family>memorydb_redis6</Family>  
      <Description>My first parameter group</Description>  
      <ARN>arn:aws:memorydb:us-east-1:012345678912:parametergroup/myredis6x</ARN>  
    </ParameterGroup>  
  </CreateParameterGroupResult>  
  <ResponseMetadata>  
    <RequestId>d8465952-af48-11e0-8d36-859edca6f4b8</RequestId>  
  </ResponseMetadata>  
</CreateParameterGroupResponse>
```

Ketika grup parameter dibuat, grup parameter ini akan memiliki nilai default keluarga. Untuk mengubah nilai default, Anda harus mengubah grup parameter. Untuk informasi selengkapnya, lihat [Mengubah grup parameter](#).

Untuk informasi selengkapnya, lihat [CreateParameterGroup](#).

Membuat daftar grup parameter berdasarkan nama

Anda dapat membuat daftar grup parameter menggunakan konsol MemoryDB, API AWS CLI, atau MemoryDB.

Menampilkan daftar grup parameter berdasarkan nama (Konsol)

Prosedur berikut menunjukkan cara melihat daftar grup parameter menggunakan konsol MemoryDB.

Untuk daftar grup parameter menggunakan konsol MemoryDB

1. [Masuk ke AWS Management Console dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/.](https://console.aws.amazon.com/memorydb/)
2. Untuk melihat daftar semua grup parameter yang tersedia, pilih Grup Parameter di panel navigasi sebelah kiri.

Daftar grup parameter berdasarkan nama (AWS CLI)

Untuk menghasilkan daftar kelompok parameter menggunakan AWS CLI, gunakan perintah `describe-parameter-groups`. Jika Anda memberikan nama grup parameter, hanya grup parameter tersebut yang akan dicantumkan. Jika Anda memberikan nama grup parameter, hanya grup parameter hingga `--max-results` yang akan dicantumkan. Dalam kedua kasus, nama, keluarga, dan deskripsi grup parameter akan dicantumkan.

Example

Kode contoh berikut mencantumkan kelompok parameter `MyRedis6x`.

Untuk Linux, macOS, atau Unix:

```
aws memorydb describe-parameter-groups \  
  --parameter-group-name myRedis6x
```

Untuk Windows:

```
aws memorydb describe-parameter-groups ^  
  --parameter-group-name myRedis6x
```

Output dari perintah ini akan terlihat seperti ini, mencantumkan nama, keluarga, dan deskripsi untuk grup parameter.


```
{
  "ParameterGroups": [
    {
      "Name": "myRedis6x",
      "Family": "memorydb_redis6",
      "Description": "My first parameter group",
      "ARN": "arn:aws:memorydb:us-east-1:012345678912:parametergroup/
myredis6x"
    }
  ]
}
```

Example

Kode contoh berikut mencantumkan grup parameter MyRedis6x untuk grup parameter yang berjalan pada mesin Redis OSS versi 5.0.6 dan seterusnya.

Untuk Linux, macOS, atau Unix:

```
aws memorydb describe-parameter-groups \
  --parameter-group-name myRedis6x
```

Untuk Windows:

```
aws memorydb describe-parameter-groups ^
  --parameter-group-name myRedis6x
```

Output dari perintah ini akan terlihat seperti ini, mencantumkan nama, keluarga, dan deskripsi untuk grup parameter.

```
{
  "ParameterGroups": [
    {
      "Name": "myRedis6x",
      "Family": "memorydb_redis6",
      "Description": "My first parameter group",
      "ARN": "arn:aws:memorydb:us-east-1:012345678912:parametergroup/
myredis6x"
    }
  ]
}
```

Example

Kode contoh berikut mencantumkan hingga 20 kelompok parameter.

```
aws memorydb describe-parameter-groups --max-results 20
```

Output JSON dari perintah ini akan terlihat seperti ini, mencantumkan nama, keluarga, dan deskripsi untuk setiap grup parameter.

```
{
  "ParameterGroups": [
    {
      "ParameterGroupName": "default.memorydb-redis6",
      "Family": "memorydb_redis6",
      "Description": "Default parameter group for memorydb_redis6",
      "ARN": "arn:aws:memorydb:us-east-1:012345678912:parametergroup/default.memorydb-redis6"
    },
    ...
  ]
}
```

Untuk informasi selengkapnya, lihat [describe-parameter-groups](#).

Daftar grup parameter berdasarkan nama (MemoryDB API)

Untuk menghasilkan daftar grup parameter menggunakan API MemoryDB, gunakan tindakan `DescribeParameterGroups` Jika Anda memberikan nama grup parameter, hanya grup parameter tersebut yang akan dicantumkan. Jika Anda memberikan nama grup parameter, hanya grup parameter hingga `MaxResults` yang akan dicantumkan. Dalam kedua kasus, nama, keluarga, dan deskripsi grup parameter akan dicantumkan.

Example

Kode contoh berikut mencantumkan hingga 20 kelompok parameter.

```
https://memory-db.us-east-1.amazonaws.com/
?Action=DescribeParameterGroups
&MaxResults=20
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20210802T192317Z
```

```
&Version=2021-01-01
&X-Amz-Credential=<credential>
```

Respons dari tindakan ini akan terlihat seperti ini, mencantumkan nama, keluarga, dan deskripsi dalam kasus `memorydb_redis6`, untuk setiap grup parameter.

```
<DescribeParameterGroupsResponse xmlns="http://memory-db.us-east-1.amazonaws.com/doc/2021-01-01/">
  <DescribeParameterGroupsResult>
    <ParameterGroups>
      <ParameterGroup>
        <Name>myRedis6x</Name>
        <Family>memorydb_redis6</Family>
        <Description>My custom Redis OSS 6 parameter group</Description>
        <ARN>arn:aws:memorydb:us-east-1:012345678912:parametergroup/myredis6x</ARN>
      </ParameterGroup>
      <ParameterGroup>
        <Name>default.memorydb-redis6</Name>
        <Family>memorydb_redis6</Family>
        <Description>Default parameter group for memorydb_redis6</Description>
        <ARN>arn:aws:memorydb:us-east-1:012345678912:parametergroup/default.memorydb-redis6</ARN>
      </ParameterGroup>
    </ParameterGroups>
  </DescribeParameterGroupsResult>
  <ResponseMetadata>
    <RequestId>3540cc3d-af48-11e0-97f9-279771c4477e</RequestId>
  </ResponseMetadata>
</DescribeParameterGroupsResponse>
```

Example

Kode contoh berikut mencantumkan kelompok parameter `MyRedis6x`.

```
https://memory-db.us-east-1.amazonaws.com/
?Action=DescribeParameterGroups
&ParameterGroupName=myRedis6x
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20210802T192317Z
&Version=2021-01-01
&X-Amz-Credential=<credential>
```

Respons dari tindakan ini akan terlihat seperti ini, menampilkan daftar nama, keluarga, dan deskripsi.

```
<DescribeParameterGroupsResponse xmlns="http://memory-db.us-east-1.amazonaws.com/doc/2021-01-01/">
  <DescribeParameterGroupsResult>
    <ParameterGroups>
      <ParameterGroup>
        <Name>myRedis6x</Name>
        <Family>memorydb_redis6</Family>
        <Description>My custom Redis OSS 6 parameter group</Description>
        <ARN>arn:aws:memorydb:us-east-1:012345678912:parametergroup/myredis6x</ARN>
      </ParameterGroup>
    </ParameterGroups>
  </DescribeParameterGroupsResult>
  <ResponseMetadata>
    <RequestId>3540cc3d-af48-11e0-97f9-279771c4477e</RequestId>
  </ResponseMetadata>
</DescribeParameterGroupsResponse>
```

Untuk informasi selengkapnya, lihat [DescribeParameterGroups](#).

Membuat daftar nilai grup parameter

Anda dapat mencantumkan parameter dan nilainya untuk grup parameter menggunakan konsol MemoryDB, API AWS CLI, atau MemoryDB.

Menampilkan daftar nilai grup parameter (Konsol)

Prosedur berikut menunjukkan cara membuat daftar parameter dan nilainya untuk grup parameter menggunakan konsol MemoryDB.

Untuk membuat daftar parameter grup parameter dan nilainya menggunakan konsol MemoryDB

1. [Masuk ke AWS Management Console dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/.](https://console.aws.amazon.com/memorydb/)
2. Untuk melihat daftar semua grup parameter yang tersedia, pilih Grup Parameter di panel navigasi sebelah kiri.
3. Pilih grup parameter yang ingin Anda cantumkan parameter dan nilainya dengan memilih nama (bukan kotak di sebelahnya) dari nama grup parameter.

Parameter dan nilainya akan tercantum di bagian bawah layar. Karena jumlah parameter, Anda mungkin harus menggulir ke atas/bawah untuk menemukan parameter yang diinginkan.

Daftar nilai kelompok parameter (AWS CLI)

Untuk membuat daftar parameter grup parameter dan nilainya menggunakan AWS CLI, gunakan perintah `describe-parameters`.

Example

Kode contoh berikut mencantumkan semua parameter dan nilainya untuk kelompok parameter `MyRedis6x`.

Untuk Linux, macOS, atau Unix:

```
aws memorydb describe-parameters \  
  --parameter-group-name myRedis6x
```

Untuk Windows:

```
aws memorydb describe-parameters ^  
  --parameter-group-name myRedis6x
```

Untuk informasi selengkapnya, lihat [describe-parameters](#).

Daftar nilai grup parameter (MemoryDB API)

Untuk membuat daftar parameter grup parameter dan nilainya menggunakan API MemoryDB, gunakan tindakan `DescribeParameters`.

Untuk informasi selengkapnya, lihat [DescribeParameters](#).

Mengubah grup parameter

Important

Anda tidak dapat mengubah grup parameter default.

Anda dapat mengubah beberapa nilai parameter di grup parameter. Nilai parameter ini diterapkan ke kluster yang terkait dengan grup parameter. Untuk informasi selengkapnya tentang kapan perubahan nilai parameter diterapkan ke grup parameter, lihat [Parameter spesifik Redis OSS](#).

Mengubah grup parameter (Konsol)

Prosedur berikut menunjukkan cara mengubah nilai parameter menggunakan konsol MemoryDB. Anda akan menggunakan prosedur yang sama untuk mengubah nilai parameter apa pun.

Untuk mengubah nilai parameter menggunakan konsol MemoryDB

1. [Masuk ke AWS Management Console dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/.](https://console.aws.amazon.com/memorydb/)
2. Untuk melihat daftar semua grup parameter yang tersedia, pilih Grup Parameter di panel navigasi sebelah kiri.
3. Pilih grup parameter yang ingin Anda modifikasi dengan memilih tombol radio di sebelah kiri nama grup parameter.

Pilih Tindakan dan kemudian Lihat detail. Atau, Anda juga dapat memilih nama grup parameter untuk pergi ke halaman detail.

4. Untuk mengubah parameter, pilih Edit. Semua parameter yang dapat diedit akan diaktifkan untuk diedit. Anda mungkin harus bergerak melintasi halaman untuk menemukan parameter yang ingin Anda ubah. Atau, Anda dapat mencari parameter berdasarkan nama, nilai, atau jenis di kotak pencarian.
5. Buat modifikasi parameter yang diperlukan.
6. Untuk menyimpan perubahan Anda, memilih Simpan perubahan.
7. Jika Anda mengubah nilai parameter di seluruh jumlah halaman, Anda dapat meninjau semua perubahan dengan memilih Pratinjau perubahan. Untuk mengonfirmasi perubahan, pilih Simpan perubahan. Untuk membuat lebih banyak modifikasi, pilih kembali.
8. Halaman detail Parameter juga memberi Anda opsi untuk mengatur ulang ke nilai default. Untuk mengatur ulang ke nilai default, pilih Reset ke default. Kotak centang akan muncul di sisi kiri semua parameter. Anda dapat memilih yang ingin Anda atur ulang dan memilih Lanjutkan untuk mengatur ulang untuk mengonfirmasi.

Pilih konfirmasi untuk mengonfirmasi tindakan reset pada kotak dialog.

9. Halaman detail parameter memungkinkan Anda mengatur jumlah parameter yang ingin Anda lihat di setiap halaman. Gunakan roda gigi di sisi kanan untuk membuat perubahan itu. Anda juga dapat mengaktifkan/menonaktifkan kolom yang Anda inginkan di halaman detail. Perubahan ini berlangsung selama sesi konsol.

Untuk menemukan nama parameter yang Anda ubah, lihat [Parameter spesifik Redis OSS](#).

Memodifikasi grup parameter (AWS CLI)

Untuk mengubah nilai parameter menggunakan AWS CLI, gunakan perintah `update-parameter-group`.

Untuk menemukan nama dan nilai yang diizinkan dari parameter yang ingin Anda ubah, lihat [Parameter spesifik Redis OSS](#)

Untuk informasi lebih lanjut, lihat [update-parameter-group](#).

Memodifikasi grup parameter (MemoryDB API)

Untuk mengubah nilai parameter grup parameter menggunakan API MemoryDB, gunakan tindakan `UpdateParameterGroup`

Untuk menemukan nama dan nilai yang diizinkan dari parameter yang ingin Anda ubah, lihat [Parameter spesifik Redis OSS](#)

Untuk informasi selengkapnya, lihat [UpdateParameterGroup](#).

Menghapus grup parameter

Anda dapat menghapus grup parameter kustom menggunakan konsol MemoryDB, API AWS CLI, atau MemoryDB.

Anda tidak dapat menghapus grup parameter jika grup parameter ini dikaitkan dengan klaster. Anda juga tidak dapat menghapus grup parameter default.

Menghapus grup parameter (Konsol)

Prosedur berikut menunjukkan cara menghapus grup parameter menggunakan konsol MemoryDB.

Untuk menghapus grup parameter menggunakan konsol MemoryDB

1. [Masuk ke AWS Management Console dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/.](https://console.aws.amazon.com/memorydb/)
2. Untuk melihat daftar semua grup parameter yang tersedia, pilih Grup Parameter di panel navigasi sebelah kiri.
3. Pilih grup parameter yang ingin Anda hapus dengan memilih tombol radio di sebelah kiri nama grup parameter.

Pilih Tindakan dan kemudian pilih Hapus.
4. Layar konfirmasi Hapus Grup Parameter akan muncul.
5. Untuk menghapus grup parameter, masukkan Hapus di kotak teks konfirmasi.

Untuk mempertahankan grup parameter, pilih Batalkan.

Menghapus grup parameter (AWS CLI)

Untuk menghapus grup parameter menggunakan AWS CLI, gunakan perintah `delete-parameter-group`. Untuk grup parameter yang akan dihapus, grup parameter yang ditentukan berdasarkan `--parameter-group-name` tidak boleh memiliki klaster yang terkait dengannya, dan juga tidak dapat berupa grup parameter default.

Kode contoh berikut menghapus grup parameter MyRedis6x.

Example

Untuk Linux, macOS, atau Unix:

```
aws memorydb delete-parameter-group \  
  --parameter-group-name myRedis6x
```

Untuk Windows:

```
aws memorydb delete-parameter-group ^  
  --parameter-group-name myRedis6x
```

Untuk informasi lebih lanjut, lihat [delete-parameter-group](#).

Menghapus grup parameter (MemoryDB API)

Untuk menghapus grup parameter menggunakan API MemoryDB, gunakan tindakan `DeleteParameterGroup` Untuk grup parameter yang akan dihapus, grup parameter yang ditentukan berdasarkan `ParameterGroupName` tidak boleh memiliki klaster yang terkait dengannya, dan juga tidak dapat berupa grup parameter default.

Example

Kode contoh berikut menghapus grup parameter `MyRedis6x`.

```
https://memory-db.us-east-1.amazonaws.com/  
?Action=DeleteParameterGroup  
&ParameterGroupName=myRedis6x  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20210802T192317Z  
&Version=2021-01-01  
&X-Amz-Credential=<credential>
```

Untuk informasi selengkapnya, lihat [DeleteParameterGroup](#).

Parameter spesifik Redis OSS

Jika Anda tidak menentukan grup parameter untuk cluster Redis OSS Anda, maka grup parameter default yang sesuai dengan versi mesin Anda akan digunakan. Anda tidak dapat mengubah nilai parameter dalam grup parameter default. Namun, Anda dapat membuat grup parameter kustom dan menentukannya ke klaster Anda setiap saat asalkan nilai parameter yang dapat diubah secara bersyarat di kedua grup parameter sama. Untuk informasi selengkapnya, lihat [Membuat grup parameter](#).

Topik

- [Redis OSS 7 perubahan parameter](#)
- [Redis OSS 6 parameter](#)
- [Parameter spesifik tipe node MemoryDB](#)

Redis OSS 7 perubahan parameter

Note

MemoryDB telah memperkenalkan rilis pratinjau [pencarian Vektor](#) yang mencakup grup parameter baru yang tidak dapat diubah. `default.memorydb-redis7.search.preview` Grup parameter ini tersedia di konsol MemoryDB dan saat membuat `vector-search-enabled` cluster baru menggunakan perintah CLI [create-cluster](#). Rilis pratinjau tersedia di AWS Wilayah berikut: US East (Virginia N.), US East (Ohio), US West (Oregon), Asia Pasifik (Tokyo), dan Eropa (Irlandia).

Keluarga kelompok parameter: `memorydb_redis7`

Parameter yang ditambahkan dalam Redis OSS 7 adalah sebagai berikut.

Nama	Detail	Deskripsi
<code>latency-tracking</code>	<p>Nilai yang diizinkan: <code>yes</code>, <code>no</code></p> <p>Default: <code>no</code></p> <p>Jenis: <code>string</code></p>	<p>Ketika diatur ke <code>ya</code> akan melacak latensi per perintah dan memungkinkan ekspor distribusi persentil melalui perintah statistik latensi <code>INFO</code>, dan distribusi latensi kumulatif (histogram) melalui perintah <code>LATENCY</code>.</p>

Nama	Detail	Deskripsi
	<p>Dapat diubah: Ya</p> <p>Penerapan perubahan: Segera di semua simpul dalam kluster.</p>	
<p>hash-max-listpack-entries</p>	<p>Nilai yang diizinkan: 0+</p> <p>Default: 512</p> <p>Jenis: integer</p> <p>Dapat diubah: Ya</p> <p>Penerapan perubahan: Segera di semua simpul dalam kluster.</p>	<p>Jumlah maksimum entri hash agar set data dikompresi.</p>
<p>hash-max-listpack-value</p>	<p>Nilai yang diizinkan: 0+</p> <p>Default: 64</p> <p>Jenis: integer</p> <p>Dapat diubah: Ya</p> <p>Penerapan perubahan: Segera di semua simpul dalam kluster.</p>	<p>Ambang entri hash terbesar agar set data dikompresi.</p>

Nama	Detail	Deskripsi
zset-max-listpack-entries	<p>Nilai yang diizinkan: 0+</p> <p>Default: 128</p> <p>Jenis: integer</p> <p>Dapat diubah: Ya</p> <p>Penerapan perubahan: Segera di semua simpul dalam klaster.</p>	<p>Jumlah maksimum entri sorted set agar set data dikompresi.</p>
zset-max-listpack-value	<p>Nilai yang diizinkan: 0+</p> <p>Default: 64</p> <p>Jenis: integer</p> <p>Dapat diubah: Ya</p> <p>Penerapan perubahan: Segera di semua simpul dalam klaster.</p>	<p>Ambang batas entri sorted set terbesar agar set data dikompresi.</p>
search-enabled	<p>Nilai yang diizinkan: yes, no</p> <p>Default: no</p> <p>Jenis: string</p> <p>Dapat diubah: Ya</p> <p>Perubahan berlaku: Hanya untuk cluster baru.</p> <p>Versi mesin minimum: 7.1</p>	<p>Ketika diatur ke ya, ini memungkinkan kemampuan Pencarian.</p>

Nama	Detail	Deskripsi
search-query-timeout-ms	<p>Nilai yang diizinkan: 1 - 60,000</p> <p>Default: 10,000</p> <p>Jenis: integer</p> <p>Dapat diubah: Ya</p> <p>Penerapan perubahan: Segera di semua simpul dalam kluster.</p> <p>Versi mesin minimum: 7.1</p>	Jumlah waktu maksimum dalam milidetik yang diizinkan untuk dijalankan oleh kueri penelusuran.

Parameter yang diubah dalam Redis OSS 7 adalah sebagai berikut.

Nama	Detail	Deskripsi
activeresharding	<p>Dapat diubah: no. Di Redis OSS 7, parameter ini disembunyikan dan diaktifkan secara default. Untuk menonaktifkannya, Anda perlu membuat kasus dukungan.</p>	Dapat diubah sebelumnya adalah ya.

Parameter dihapus dalam Redis OSS 7 adalah sebagai berikut.

Nama	Detail	Deskripsi
hash-max-ziplist-entries	<p>Nilai yang diizinkan: 0+</p> <p>Default: 512</p>	Gunakan listpack bukan ziplist untuk merepresentasikan pengkodean hash kecil.

Nama	Detail	Deskripsi
	<p>Jenis: integer</p> <p>Dapat diubah: Ya</p> <p>Penerapan perubahan: Segera di semua simpul dalam klaster.</p>	
<p>hash-max-ziplist-value</p>	<p>Nilai yang diizinkan: 0+</p> <p>Default: 64</p> <p>Jenis: integer</p> <p>Dapat diubah: Ya</p> <p>Penerapan perubahan: Segera di semua simpul dalam klaster.</p>	<p>Gunakan listpack bukan ziplist untuk merepresentasikan pengkodean hash kecil.</p>
<p>zset-max-ziplist-entries</p>	<p>Nilai yang diizinkan: 0+</p> <p>Default: 128</p> <p>Jenis: integer</p> <p>Dapat diubah: Ya</p> <p>Penerapan perubahan: Segera di semua simpul dalam klaster.</p>	<p>Gunakan listpack bukan ziplist untuk merepresentasikan pengkodean hash kecil.</p>

Nama	Detail	Deskripsi
zset-max-ziplist-value	<p>Nilai yang diizinkan: 0+</p> <p>Default: 64</p> <p>Jenis: integer</p> <p>Dapat diubah: Ya</p> <p>Penerapan perubahan: Segera di semua simpul dalam kluster.</p>	Gunakan listpack bukan ziplist untuk merepresentasikan pengkodean hash kecil.

Redis OSS 6 parameter

Note

Di mesin Redis OSS versi 6.2, ketika keluarga simpul r6gd diperkenalkan untuk digunakan dengan [Tingkatan data](#), hanyanoeviction, volatile-lru dan kebijakan allkeys-lru memori maksimum didukung dengan tipe node r6gd.

Keluarga kelompok parameter: memorydb_redis6

Parameter yang ditambahkan dalam Redis OSS 6 adalah sebagai berikut.

Nama	Detail	Deskripsi
maxmemory-policy	<p>Jenis: STRING</p> <p>Nilai yang diizinkan: volatile-lru, allkeys-lru, volatile-lfu, allkeys-lfu, volatile-random, allkeys-random, volatile-ttl, noeviction</p>	<p>Kebijakan pengosongan untuk kunci saat penggunaan memori maksimum tercapai.</p> <p>Untuk informasi selengkapnya, lihat Menggunakan Redis OSS sebagai cache LRU.</p>

Nama	Detail	Deskripsi
	Default: noeviction	
list-compress-depth	Jenis: INTEGER Nilai yang diizinkan: 0- Default: 0	<p>Kedalaman kompresi adalah jumlah simpul quicklist ziplist dari setiap sisi daftar yang akan dikecualikan dari kompresi. Kepala dan ekor dari daftar selalu tidak dikompresi untuk operasi "fast push and pop". Pengaturannya adalah:</p> <ul style="list-style-type: none"> • 0: Menonaktifkan semua kompresi. • 1: Mulai mengompresi dengan simpul pertama masuk dari kepala dan ekor. [head]->node->node->...->node->[tail] <p>Semua simpul kecuali jika [head] dan [tail] dikompresi.</p> <ul style="list-style-type: none"> • 2: Mulai mengompresi dengan simpul kedua masuk dari kepala dan ekor. [head]->[next]->node->node->...->node->[prev]->[tail] <p>[head], [next], [prev], [tail] tidak dikompresi. Semua simpul lainnya dikompresi.</p> <ul style="list-style-type: none"> • Dll.

Nama	Detail	Deskripsi
hll-sparse-max-bytes	<p>Jenis: INTEGER</p> <p>Nilai yang diizinkan: 1-16000</p> <p>Default: 3000</p>	<p>HyperLogLog batas byte representasi jarang. Batas termasuk header 16 byte. Ketika HyperLogLog menggunakan representasi jarang melintasi batas ini, itu diubah menjadi representasi padat.</p> <p>Nilai yang lebih besar dari 16000 tidak disarankan karena pada titik tersebut dense representation lebih hemat memori.</p> <p>Kami merekomendasikan nilai sekitar 3000 untuk mendapatkan manfaat dari pengkodean hemat ruang tanpa memperlambat PFADD terlalu banyak, yaitu $O(N)$ dengan pengkodean yang jarang. Nilai dapat dinaikkan menjadi ~ 10000 ketika CPU tidak menjadi perhatian, tetapi ruang adalah, dan kumpulan data terdiri dari banyak HyperLogLogs dengan kardinalitas dalam kisaran 0 - 15000.</p>
lfu-log-factor	<p>Jenis: INTEGER</p> <p>Nilai yang diizinkan: 1-</p> <p>Default: 10</p>	<p>Faktor log untuk menambah penghitung kunci untuk kebijakan pengusuran LFU.</p>
lfu-decay-time	<p>Jenis: INTEGER</p> <p>Nilai yang diizinkan: 0-</p> <p>Default: 1</p>	<p>Jumlah waktu dalam hitungan menit untuk mengurangi penghitung kunci untuk kebijakan pengusuran LFU.</p>

Nama	Detail	Deskripsi
<code>active-defrag-max-scan-fields</code>	<p>Jenis: INTEGER</p> <p>Nilai yang diizinkan: 1-1000000</p> <p>Default: 1000</p>	Jumlah maksimum bidang set/hash/zset/list yang akan diproses dari pemindaian kamus utama selama defragmentasi aktif.
<code>active-defrag-threshold-upper</code>	<p>Jenis: INTEGER</p> <p>Nilai yang diizinkan: 1-100</p> <p>Default: 100</p>	Persentase maksimum fragmentasi yang mana kita menggunakan upaya maksimal.
<code>client-output-buffer-limit-pubsub-hard-limit</code>	<p>Jenis: INTEGER</p> <p>Nilai yang diizinkan: 0-</p> <p>Default: 33554432</p>	Untuk klien penerbitan/berlangganan Redis OSS: Jika buffer keluaran klien mencapai jumlah byte yang ditentukan, klien akan terputus.
<code>client-output-buffer-limit-pubsub-soft-limit</code>	<p>Jenis: INTEGER</p> <p>Nilai yang diizinkan: 0-</p> <p>Default: 8388608</p>	Untuk klien terbitkan/berlangganan Redis OSS: Jika buffer keluaran klien mencapai jumlah byte yang ditentukan, klien akan terputus, tetapi hanya jika kondisi ini berlanjut <code>client-output-buffer-limit-pubsub-soft-seconds</code> .
<code>client-output-buffer-limit-pubsub-soft-seconds</code>	<p>Jenis: INTEGER</p> <p>Nilai yang diizinkan: 0-</p> <p>Default: 60</p>	Untuk klien penerbitan/berlangganan Redis OSS: Jika buffer keluaran klien tetap pada <code>client-output-buffer-limit-pubsub-soft-limit</code> byte lebih lama dari jumlah detik ini, klien akan terputus.

Nama	Detail	Deskripsi
timeout	<p>Jenis: INTEGER</p> <p>Nilai yang diizinkan: 0,20-</p> <p>Default: 0</p>	<p>Jumlah detik waktu tunggu simpul sebelum waktu habis Nilainya adalah:</p> <ul style="list-style-type: none"> • 0 - jangan pernah memutuskan klien yang menganggur. • 1-19 - nilai tidak valid. • >=20 — jumlah detik node menunggu sebelum memutuskan sambungan klien idle.
notify-keyspace-events	<p>Jenis: STRING</p> <p>Nilai yang diizinkan: NULL</p> <p>Default: NULL</p>	<p>Acara keyspace untuk Redis OSS untuk memberi tahu klien Pub/Sub tentang. Secara default semua notifikasi dinonaktifkan.</p>
maxmemory-samples	<p>Jenis: INTEGER</p> <p>Nilai yang diizinkan: 1-</p> <p>Default: 3</p>	<p>Untuk least-recently-used (LRU) dan time-to-live (TTL) perhitungan, parameter ini mewakili ukuran sampel kunci untuk diperiksa. Secara default, Redis OSS memilih 3 kunci dan menggunakan salah satu yang paling tidak digunakan baru-baru ini.</p>
slowlog-max-len	<p>Jenis: INTEGER</p> <p>Nilai yang diizinkan: 0-</p> <p>Default: 128</p>	<p>Panjang maksimum Redis OSS Slow Log. Tidak ada batasan untuk panjang ini. Ketahuilah bahwa itu akan menghabiskan memori. Anda dapat merebut kembali memori yang digunakan oleh log lambat dengan SLOWLOG RESET.</p>

Nama	Detail	Deskripsi
<code>activereshashing</code>	<p>Jenis: STRING</p> <p>Nilai yang diizinkan: ya, tidak</p> <p>Default: yes</p>	<p>Tabel hash utama di-rehash sepuluh kali per detik; setiap operasi rehash mengonsumsi 1 milidetik waktu CPU.</p> <p>Nilai ini diatur saat Anda membuat grup parameter. Ketika menetapkan grup parameter baru untuk klaster, nilai ini harus sama baik dalam grup parameter lama dan baru.</p>
<code>client-output-buffer-limit-normal-hard-limit</code>	<p>Jenis: INTEGER</p> <p>Nilai yang diizinkan: 0-</p> <p>Default: 0</p>	<p>Jika buffer output klien mencapai jumlah byte tertentu, klien akan terputus. Default-nya adalah nol (tidak ada batas absolut).</p>
<code>client-output-buffer-limit-normal-soft-limit</code>	<p>Jenis: INTEGER</p> <p>Nilai yang diizinkan: 0-</p> <p>Default: 0</p>	<p>Jika buffer output klien mencapai jumlah byte tertentu, klien akan terputus, tetapi hanya jika kondisi ini bertahan selama <code>client-output-buffer-limit-normal-soft-seconds</code>. Default-nya adalah nol (tidak ada batas relatif).</p>
<code>client-output-buffer-limit-normal-soft-seconds</code>	<p>Jenis: INTEGER</p> <p>Nilai yang diizinkan: 0-</p> <p>Default: 0</p>	<p>Jika buffer output klien tetap pada <code>client-output-buffer-limit-normal-soft-limit</code> byte lebih lama dari jumlah detik ini, klien akan terputus. Default-nya adalah nol (tidak ada batas waktu).</p>
<code>tcp-keepalive</code>	<p>Jenis: INTEGER</p> <p>Nilai yang diizinkan: 0-</p> <p>Default: 300</p>	<p>Jika parameter ini diatur ke nilai bukan nol (N), simpul klien akan di-polling setiap N detik untuk memastikan bahwa simpul ini masih terhubung. Dengan pengaturan default 0, tidak ada polling yang terjadi.</p>

Nama	Detail	Deskripsi
active-defrag-cycle-min	Jenis: INTEGER Nilai yang diizinkan: 1-75 Default: 5	Upaya minimal untuk defrag dalam persentase CPU.
stream-node-max-bytes	Jenis: INTEGER Nilai yang diizinkan: 0- Default: 4096	Struktur aliran data adalah pohon radix simpul yang mengodekan beberapa item dalamnya. Gunakan konfigurasi ini untuk menentukan ukuran maksimum simpul tunggal dalam pohon radix dalam Byte. Jika diatur ke 0, ukuran simpul pohon adalah tidak terbatas.
stream-node-max-entries	Jenis: INTEGER Nilai yang diizinkan: 0- Default: 100	Struktur data aliran adalah pohon radix simpul yang mengkode beberapa item dalamnya. Gunakan konfigurasi ini untuk menentukan jumlah maksimum item yang dapat ditampung simpul tunggal sebelum beralih ke simpul baru saat menambahkan entri aliran baru. Jika disetel ke 0, jumlah item di simpul pohon tidak terbatas.
lazyfree-lazy- eviction	Jenis: STRING Nilai yang diizinkan: ya, tidak Default: no	Lakukan penghapusan asinkron pada pengusuran.
active-defrag-ignore-bytes	Jenis: INTEGER Nilai yang diizinkan: 1048576- Default: 104857600	Jumlah minimum sisa fragmentasi untuk memulai defrag aktif.

Nama	Detail	Deskripsi
lazyfree-lazy-expire	<p>Jenis: STRING</p> <p>Nilai yang diizinkan: ya, tidak</p> <p>Default: no</p>	Lakukan penghapusan asinkron pada kunci kedaluwarsa.
active-defrag-threshold-lower	<p>Jenis: INTEGER</p> <p>Nilai yang diizinkan: 1-100</p> <p>Default: 10</p>	Persentase minimum fragmentasi untuk memulai defrag aktif.
active-defrag-cycle-max	<p>Jenis: INTEGER</p> <p>Nilai yang diizinkan: 1-75</p> <p>Default: 75</p>	Upaya maksimal untuk defrag dalam persentase CPU.
lazyfree-lazy-server-del	<p>Jenis: STRING</p> <p>Nilai yang diizinkan: ya, tidak</p> <p>Default: no</p>	Melakukan penghapusan asinkron untuk perintah yang memperbarui nilai.
slowlog-log-slower-than	<p>Jenis: INTEGER</p> <p>Nilai yang diizinkan: 0-</p> <p>Default: 10000</p>	Waktu eksekusi maksimum, dalam mikrodetik, melebihi agar perintah dapat dicatat oleh fitur Redis Slow Log OSS. Perhatikan bahwa angka negatif menonaktifkan log lambat, sementara nilai nol memaksa pencatatan setiap perintah.

Nama	Detail	Deskripsi
<code>hash-max-ziplist-entries</code>	<p>Jenis: INTEGER</p> <p>Nilai yang diizinkan: 0-</p> <p>Default: 512</p>	Menentukan jumlah memori yang digunakan untuk hash. Hash dengan jumlah entri kurang dari yang ditentukan akan disimpan menggunakan pengodean khusus yang menghemat ruang.
<code>hash-max-ziplist-value</code>	<p>Jenis: INTEGER</p> <p>Nilai yang diizinkan: 0-</p> <p>Default: 64</p>	Menentukan jumlah memori yang digunakan untuk hash. Hash dengan entri yang lebih kecil dari jumlah byte yang ditentukan akan disimpan menggunakan pengodean khusus yang menghemat ruang.
<code>set-max-intset-entries</code>	<p>Jenis: INTEGER</p> <p>Nilai yang diizinkan: 0-</p> <p>Default: 512</p>	Menentukan jumlah memori yang digunakan untuk jenis tertentu dari set (string yang berupa integer dalam radix 10 pada rentang integer bertanda 64 bit). Set seperti itu dengan jumlah entri kurang dari yang ditentukan akan disimpan menggunakan pengodean khusus yang menghemat ruang.
<code>zset-max-ziplist-entries</code>	<p>Jenis: INTEGER</p> <p>Nilai yang diizinkan: 0-</p> <p>Default: 128</p>	Menentukan jumlah memori yang digunakan untuk set berurutan. Sorted set dengan jumlah elemen kurang dari yang ditentukan akan disimpan menggunakan pengodean khusus yang menghemat ruang.
<code>zset-max-ziplist-value</code>	<p>Jenis: INTEGER</p> <p>Nilai yang diizinkan: 0-</p> <p>Default: 64</p>	Menentukan jumlah memori yang digunakan untuk sorted set. Sorted set dengan entri yang lebih kecil dari jumlah byte yang ditentukan disimpan menggunakan pengodean khusus yang menghemat ruang.

Nama	Detail	Deskripsi
tracking-table-max-keys	Jenis: INTEGER Nilai yang diizinkan: 1-100000000 Default: 1000000	Untuk membantu caching sisi klien, Redis OSS mendukung pelacakan klien mana yang telah mengakses kunci mana. Ketika kunci yang dilacak diubah, pesan invalidasi dikirim ke semua klien untuk memberitahukan bahwa nilai cache-nya tidak valid lagi. Nilai ini memungkinkan Anda menentukan batas atas tabel ini.
acllog-max-len	Jenis: INTEGER Nilai yang diizinkan: 1-10000 Default: 128	Jumlah maksimum entri dalam Log ACL.

Nama	Detail	Deskripsi
active-expire-effort	<p>Jenis: INTEGER</p> <p>Nilai yang diizinkan: 1-10</p> <p>Default: 1</p>	<p>Redis OSS menghapus kunci yang telah melampaui waktu mereka untuk hidup dengan dua mekanisme. Di satu sisi, kunci diakses dan ditemukan akan kedaluwarsa. Di sisi lain, pekerjaan berkala mengambil sampel kunci dan membuat kunci yang telah melebihi time-to-live (TTL)-nya menjadi kedaluwarsa. Parameter ini mendefinisikan jumlah upaya yang digunakan Redis OSS untuk mengakhiri item dalam pekerjaan periodik.</p> <p>Nilai default 1 akan mencoba mencegah adanya lebih dari 10 persen kunci kedaluwarsa yang masih berada dalam memori. Hal ini juga akan mencoba mencegah konsumsi lebih dari 25 persen dari total memori dan menambahkan latensi ke sistem. Anda dapat meningkatkan nilai ini hingga 10 untuk meningkatkan jumlah upaya yang digunakan untuk kunci kedaluwarsa. Komprominya adalah CPU lebih tinggi dan latensi berpotensi lebih tinggi. Kami merekomendasikan nilai 1 kecuali jika Anda melihat penggunaan memori tinggi dan dapat menoleransi peningkatan pemanfaatan CPU.</p>
lazyfree-lazy-user-del	<p>Jenis: STRING</p> <p>Nilai yang diizinkan: ya, tidak</p> <p>Default: no</p>	<p>Menentukan apakah perilaku default DEL perintah bertindak sama seperti UNLINK.</p>

Nama	Detail	Deskripsi
<code>activedefrag</code>	<p>Jenis: STRING</p> <p>Nilai yang diizinkan: ya, tidak</p> <p>Default: no</p>	Diaktifkan defragmentasi memori aktif.
<code>maxclients</code>	<p>Jenis: INTEGER</p> <p>Nilai yang diizinkan: 65000</p> <p>Default: 65000</p>	Jumlah maksimum klien yang dapat dihubungkan pada satu waktu. Tidak dapat dimodifikasi.
<code>client-query-buffer-limit</code>	<p>Jenis: INTEGER</p> <p>Nilai yang diizinkan: 1048576-1073741824</p> <p>Default: 1073741824</p>	Ukuran maks buffer kueri klien tunggal. Perubahan terjadi segera.
<code>proto-max-bulk-len</code>	<p>Jenis: INTEGER</p> <p>Nilai yang diizinkan: 1048576-536870912</p> <p>Default: 536870912</p>	Ukuran maks dari permintaan elemen tunggal. Perubahan terjadi segera.

Parameter spesifik tipe node MemoryDB

Meskipun sebagian besar parameter memiliki nilai tunggal, beberapa parameter memiliki nilai yang berbeda-beda bergantung pada jenis simpul yang digunakan. Tabel berikut menunjukkan nilai default untuk `maxmemory` untuk setiap jenis node. Nilai `maxmemory` adalah jumlah maksimum byte yang tersedia untuk Anda gunakan, untuk data, dan untuk penggunaan lainnya, pada simpul.

Jenis simpul	Maxmemory
db.r7g.large	14037181030
db.r7g.xlarge	28261849702
db.r7g.2xlarge	56711183565
db.r7g.4xlarge	113609865216
db.r7g.8xlarge	225000375228
db.r7g.12xlarge	341206346547
db.r7g.16xlarge	450000750456
db.r6gd.xlarge	28261849702
db.r6gd.2xlarge	56711183565
db.r6gd.4xlarge	113609865216
db.r6gd.8xlarge	225000375228
db.r6g.large	14037181030
db.r6g.xlarge	28261849702
db.r6g.2xlarge	56711183565
db.r6g.4xlarge	113609865216
db.r6g.8xlarge	225000375228
db.r6g.12xlarge	341206346547
db.r6g.16xlarge	450000750456
db.t4g.small	1471026299
db.t4g.medium	3317862236

Note

Semua jenis instans MemoryDB harus dibuat dalam VPC Amazon Virtual Private Cloud.

Tutorial: Mengonfigurasi fungsi Lambda untuk mengakses MemoryDB di VPC Amazon

Dalam tutorial ini Anda dapat mempelajari cara:

- Buat klaster MemoryDB di Amazon Virtual Private Cloud (Amazon VPC) default Anda di wilayah us-east-1.
- Buat fungsi Lambda untuk mengakses klaster. Saat membuat fungsi Lambda, Anda perlu menyediakan ID subnet di Amazon VPC dan grup keamanan VPC agar fungsi Lambda dapat mengakses sumber daya di VPC Anda. Sebagai ilustrasi dalam tutorial ini, fungsi Lambda menghasilkan UUID, menuliskannya ke cluster, dan mengambilnya dari cluster..
- Panggil fungsi Lambda secara manual dan verifikasi bahwa ia mengakses cluster di VPC Anda.
- Bersihkan fungsi Lambda, klaster, dan peran IAM yang disiapkan untuk tutorial ini.

Topik

- [Langkah 1: Buat cluster](#)
- [Langkah 2: Buat fungsi Lambda](#)
- [Langkah 3: Uji fungsi Lambda](#)
- [Langkah 4: Bersihkan \(Opsional\)](#)

Langkah 1: Buat cluster

Untuk membuat cluster, ikuti langkah-langkah ini.

Topik

- [Langkah 1.1: Buat cluster](#)
- [Langkah 1.2: Salin titik akhir cluster](#)
- [Langkah 1.3: Buat Peran IAM](#)

- [Langkah 1.4: Buat Daftar Kontrol Akses \(ACL\)](#)

Langkah 1.1: Buat cluster

Pada langkah ini, Anda membuat cluster di VPC Amazon default di wilayah us-east-1 di akun Anda menggunakan (CLI). AWS Command Line Interface Untuk informasi tentang membuat klaster menggunakan konsol atau API MemoryDB, lihat. [Langkah 1: Buat cluster](#)

```
aws memorydb create-cluster --cluster-name cluster-01 --engine-version 7.0 --acl-name
open-access \
--description "MemoryDB IAM auth application" \
--node-type db.r6g.large
```

Perhatikan bahwa nilai bidang Status diatur ke CREATING. Diperlukan beberapa menit bagi MemoryDB untuk menyelesaikan pembuatan cluster Anda.

Langkah 1.2: Salin titik akhir cluster

Verifikasi bahwa MemoryDB telah selesai membuat cluster dengan perintah. `describe-clusters`

```
aws memorydb describe-clusters \
--cluster-name cluster-01
```

Salin Alamat Titik Akhir Cluster yang ditunjukkan pada output. Anda akan memerlukan alamat ini saat membuat paket deployment untuk fungsi Lambda Anda.

Langkah 1.3: Buat Peran IAM

1. Buat dokumen kebijakan kepercayaan IAM, seperti yang ditunjukkan di bawah ini, untuk peran Anda yang memungkinkan akun Anda mengambil peran baru. Simpan kebijakan ini ke file bernama `trust-policy.json`. Pastikan untuk mengganti `account_id` 123456789012 dalam kebijakan ini dengan `account_id` Anda.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": { "AWS": "arn:aws:iam::123456789012:root" },
    "Action": "sts:AssumeRole"
  }],
}
```

```

    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}

```

2. Buat dokumen kebijakan IAM, seperti yang ditunjukkan di bawah ini. Simpan kebijakan ke file bernama `policy.json`. Pastikan untuk mengganti `account_id` 123456789012 dalam kebijakan ini dengan `account_id` Anda.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect" : "Allow",
      "Action" : [
        "memorydb:Connect"
      ],
      "Resource" : [
        "arn:aws:memorydb:us-east-1:123456789012:cluster/cluster-01",
        "arn:aws:memorydb:us-east-1:123456789012:user/iam-user-01"
      ]
    }
  ]
}

```

3. Buat peran IAM.

```

aws iam create-role \
  --role-name "memorydb-iam-auth-app" \
  --assume-role-policy-document file://trust-policy.json

```

4. Buat kebijakan IAM.

```

aws iam create-policy \
  --policy-name "memorydb-allow-all" \
  --policy-document file://policy.json

```

5. Lampirkan kebijakan IAM ke peran tersebut. Pastikan untuk mengganti `account_id` 123456789012 dalam kebijakan ini dengan `account_id` Anda.

```
aws iam attach-role-policy \  
  --role-name "memorydb-iam-auth-app" \  
  --policy-arn "arn:aws:iam::123456789012:policy/memorydb-allow-all"
```

Langkah 1.4: Buat Daftar Kontrol Akses (ACL)

1. Buat pengguna baru yang didukung IAM.

```
aws memorydb create-user \  
  --user-name iam-user-01 \  
  --authentication-mode Type=iam \  
  --access-string "on ~* +@all"
```

2. Buat ACL dan lampirkan ke cluster.

```
aws memorydb create-acl \  
  --acl-name iam-acl-01 \  
  --user-names iam-user-01  
  
aws memorydb update-cluster \  
  --cluster-name cluster-01 \  
  --acl-name iam-acl-01
```

Langkah 2: Buat fungsi Lambda

Untuk membuat fungsi Lambda, lakukan langkah-langkah ini.

Topik

- [Langkah 2.1: Buat paket deployment](#)
- [Langkah 2.2: Buat peran IAM \(peran eksekusi\)](#)
- [Langkah 2.3: Unggah paket deployment \(buat fungsi Lambda\)](#)

Langkah 2.1: Buat paket deployment

Dalam tutorial ini, kami memberikan contoh kode dalam Python untuk fungsi Lambda Anda.

Python

Contoh kode Python berikut membaca dan menulis item ke cluster MemoryDB Anda. Salin kode tersebut dan simpan ke dalam file bernama `app.py`. Pastikan untuk mengganti `cluster_endpoint` nilai dalam kode dengan alamat titik akhir yang Anda salin pada langkah 1.2.

```
from typing import Tuple, Union
from urllib.parse import ParseResult, urlencode, urlunparse

import boto3.session
import redis
from boto3.model import ServiceId
from boto3.signers import RequestSigner
from cachetools import TTLCache, cached
import uuid

class MemoryDBIAMProvider(redis.CredentialProvider):
    def __init__(self, user, cluster_name, region="us-east-1"):
        self.user = user
        self.cluster_name = cluster_name
        self.region = region

        session = boto3.session.get_session()
        self.request_signer = RequestSigner(
            ServiceId("memorydb"),
            self.region,
            "memorydb",
            "v4",
            session.get_credentials(),
            session.get_component("event_emitter"),
        )

    # Generated IAM tokens are valid for 15 minutes
    @cached(cache=TTLCache(maxsize=128, ttl=900))
    def get_credentials(self) -> Union[Tuple[str], Tuple[str, str]]:
        query_params = {"Action": "connect", "User": self.user}

        url = urlunparse(
            ParseResult(
                scheme="https",
                netloc=self.cluster_name,
                path="/",
                query=urlencode(query_params),
                params="",
                fragment="",
            )
        )
```

```

    )
    )
    signed_url = self.request_signer.generate_presigned_url(
        {"method": "GET", "url": url, "body": {}, "headers": {}, "context": {}},
        operation_name="connect",
        expires_in=900,
        region_name=self.region,
    )
    # RequestSigner only seems to work if the URL has a protocol, but
    # MemoryDB only accepts the URL without a protocol
    # So strip it off the signed URL before returning
    return (self.user, signed_url.removeprefix("https://"))

def lambda_handler(event, context):
    username = "iam-user-01" # replace with your user id
    cluster_name = "cluster-01" # replace with your cache name
    cluster_endpoint = "clustercfg.cluster-01.xxxxxx.memorydb.us-east-1.amazonaws.com"
    # replace with your cluster endpoint
    creds_provider = MemoryDBIAMProvider(user=username, cluster_name=cluster_name)
    redis_client = redis.Redis(host=cluster_endpoint, port=6379,
    credential_provider=creds_provider, ssl=True, ssl_cert_reqs="none")

    key='uuid'
    # create a random UUID - this will be the sample element we add to the cluster
    uuid_in = uuid.uuid4().hex
    redis_client.set(key, uuid_in)
    result = redis_client.get(key)
    decoded_result = result.decode("utf-8")
    # check the retrieved item matches the item added to the cluster and print
    # the results
    if decoded_result == uuid_in:
        print(f"Success: Inserted {uuid_in}. Fetched {decoded_result} from MemoryDB.")
    else:
        raise Exception(f"Bad value retrieved. Expected {uuid_in}, got
        {decoded_result}")

    return "Fetched value from MemoryDB"

```

Kode ini menggunakan `redis-py` pustaka Python untuk menempatkan item ke dalam cluster Anda dan mengambilnya. Kode ini digunakan `cachetools` untuk menyimpan token Auth IAM yang dihasilkan selama 15 menit. Untuk membuat paket penyebaran yang berisi `redis-py` dan `cachetools`, lakukan langkah-langkah berikut.

Di direktori proyek Anda yang berisi file kode `app.py` sumber, buat paket folder untuk menginstal `redis-py` dan `cachetools` pustaka ke dalamnya.

```
mkdir package
```

Instal `redis-py` dan `cachetools` gunakan `pip`.

```
pip install --target ./package redis
pip install --target ./package cachetools
```

Buat `file.zip` yang berisi `redis-py` dan `cachetools` pustaka. Di Linux dan macOS, jalankan perintah berikut. Di Windows, gunakan utilitas `zip` pilihan Anda untuk membuat `file.zip` dengan `redis-py` dan `cachetools` perpustakaan di root.

```
cd package
zip -r ../my_deployment_package.zip
```

Tambahkan kode fungsi Anda ke file `.zip`. Di Linux atau macOS, jalankan perintah CLI berikut. Di Windows, gunakan utilitas `zip` pilihan Anda untuk menambahkan `app.py` ke root `file.zip` Anda.

```
cd ..
zip my_deployment_package.zip app.py
```

Langkah 2.2: Buat peran IAM (peran eksekusi)

Lampirkan kebijakan AWS terkelola yang diberi nama `AWSLambdaVPCAccessExecutionRole` ke peran.

```
aws iam attach-role-policy \
  --role-name "memorydb-iam-auth-app" \
  --policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaVPCAccessExecutionRole"
```

Langkah 2.3: Unggah paket deployment (buat fungsi Lambda)

Pada langkah ini, Anda membuat fungsi Lambda (`AccessMemoryDB`) menggunakan perintah AWS CLI `create-function`.

Dari direktori proyek yang berisi berkas `.zip` paket penyebaran Anda, jalankan perintah Lambda CLI berikut. `create-function`

Untuk opsi peran, gunakan ARN dari peran eksekusi yang Anda buat di langkah 2.2. Untuk vpc-config masukkan daftar yang dipisahkan koma dari subnet VPC default Anda dan ID grup keamanan VPC default Anda. Anda dapat menemukan nilai-nilai ini di konsol Amazon VPC. Untuk menemukan subnet VPC default Anda, pilih VPC Anda, lalu pilih VPC default akun Anda AWS . Untuk menemukan grup keamanan untuk VPC ini, buka Keamanan dan pilih Grup keamanan. Pastikan Anda memilih wilayah us-east-1.

```
aws lambda create-function \  
--function-name AccessMemoryDB \  
--region us-east-1 \  
--zip-file fileb://my_deployment_package.zip \  
--role arn:aws:iam::123456789012:role/memorydb-iam-auth-app \  
--handler app.lambda_handler \  
--runtime python3.12 \  
--timeout 30 \  
--vpc-config SubnetIds=comma-separated-vpc-subnet-ids,SecurityGroupIds=default-  
security-group-id
```

Langkah 3: Uji fungsi Lambda

Pada langkah ini, Anda menjalankan fungsi Lambda secara manual menggunakan perintah pemanggilan. Ketika fungsi Lambda dijalankan, ia menghasilkan UUID dan menulisnya ke ElastiCache cache yang Anda tentukan dalam kode Lambda Anda. Fungsi Lambda kemudian mengambil item dari cache.

1. Memanggil fungsi Lambda AccessMemory (DB) menggunakan perintah AWS Lambda pemanggilan.

```
aws lambda invoke \  
--function-name AccessMemoryDB \  
--region us-east-1 \  
output.txt
```

2. Verifikasikan bahwa fungsi Lambda berhasil dijalankan sebagai berikut:
 - Tinjau file output.txt.
 - Verifikasi hasil di CloudWatch Log dengan membuka CloudWatch konsol dan memilih grup log untuk fungsi Anda (AccessRedis/aws/lambda/). Log stream akan berisi output seperti yang berikut ini:

```
Success: Inserted 826e70c5f4d2478c8c18027125a3e01e. Fetched
826e70c5f4d2478c8c18027125a3e01e from MemoryDB.
```

- Tinjau hasilnya di AWS Lambda konsol.

Langkah 4: Bersihkan (Opsional)

Untuk membersihkannya, ambil langkah-langkah ini.

Topik

- [Langkah 4.1: Hapus fungsi Lambda](#)
- [Langkah 4.2: Hapus cluster MemoryDB](#)
- [Langkah 4.3: Hapus Peran dan kebijakan IAM](#)

Langkah 4.1: Hapus fungsi Lambda

```
aws lambda delete-function \  
--function-name AccessMemoryDB
```

Langkah 4.2: Hapus cluster MemoryDB

Hapus klaster .

```
aws memorydb delete-cluster \  
--cluster-name cluster-01
```

Hapus pengguna dan ACL.

```
aws memorydb delete-user \  
--user-id iam-user-01  
  
aws memorydb delete-acl \  
--acl-name iam-acl-01
```

Langkah 4.3: Hapus Peran dan kebijakan IAM

```
aws iam detach-role-policy \  

```

```
--role-name "memorydb-iam-auth-app" \  
--policy-arn "arn:aws:iam::123456789012:policy/memorydb-allow-all"  
  
aws iam detach-role-policy \  
--role-name "memorydb-iam-auth-app" \  
--policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaVPCAccessExecutionRole"  
  
aws iam delete-role \  
--role-name "memorydb-iam-auth-app"  
  
aws iam delete-policy \  
--policy-arn "arn:aws:iam::123456789012:policy/memorydb-allow-all"
```

Pencarian vektor

Pencarian vektor untuk MemoryDB memperluas fungsionalitas MemoryDB. Pencarian vektor dapat digunakan bersama dengan fungsionalitas MemoryDB yang ada. Aplikasi yang tidak menggunakan pencarian vektor tidak terpengaruh oleh keberadaannya. Pencarian vektor tersedia di semua Wilayah yang MemoryDB tersedia.

Pencarian vektor untuk menyederhanakan arsitektur aplikasi Anda sambil memberikan pencarian vektor berkecepatan tinggi. Pencarian vektor untuk MemoryDB sangat ideal untuk kasus penggunaan di mana kinerja puncak dan skala adalah kriteria pemilihan yang paling penting. Anda dapat menggunakan data MemoryDB atau Redis yang ada OSS API untuk membangun pembelajaran mesin dan kasus penggunaan AI generatif, seperti generasi tambahan pengambilan, deteksi anomali, pengambilan dokumen, dan rekomendasi waktu nyata.

Pada 26/6/2024, AWS MemoryDB memberikan kinerja pencarian vektor tercepat dengan tingkat penarikan tertinggi di antara database vektor populer di AWS

Topik

- [Ikhtisar pencarian vektor](#)
- [Kasus penggunaan](#)
- [Fitur pencarian vektor dan batas](#)
- [Menggunakan AWS Management Console](#)
- [Menggunakan AWS Command Line Interface](#)
- [Perintah pencarian vektor](#)

Ikhtisar pencarian vektor

Pencarian vektor dibangun di atas pembuatan, pemeliharaan, dan penggunaan indeks. Setiap operasi pencarian vektor menentukan indeks tunggal dan operasinya terbatas pada indeks itu, yaitu, operasi pada satu indeks tidak terpengaruh oleh operasi pada indeks lainnya. Kecuali untuk operasi untuk membuat dan menghancurkan indeks, sejumlah operasi dapat dikeluarkan terhadap indeks apa pun kapan saja, yang berarti bahwa pada tingkat cluster, beberapa operasi terhadap beberapa indeks dapat berlangsung secara bersamaan.

Indeks individu diberi nama objek yang ada di namespace unik, yang terpisah dari ruang nama Redis lainnya: kunci, OSS fungsi, dll. Setiap indeks secara konseptual mirip dengan tabel database

konvensional karena terstruktur dalam dua dimensi: kolom dan baris. Setiap baris dalam tabel sesuai dengan Redis OSS Key. Setiap kolom dalam indeks sesuai dengan anggota atau bagian dari kunci itu. Dalam dokumen ini, kunci istilah, baris dan catatan identik dan digunakan secara bergantian. Demikian pula istilah kolom, bidang, jalur dan anggota pada dasarnya identik dan juga digunakan secara bergantian.

Tidak ada perintah khusus untuk menambah, menghapus, atau memodifikasi data yang diindeks. Sebaliknya yang ada HASH atau JSON perintah yang memodifikasi kunci yang ada dalam indeks juga secara otomatis memperbarui indeks.

Topik

- [Indeks dan ruang kunci Redis OSS](#)
- [Jenis bidang indeks](#)
- [Algoritma indeks vektor](#)
- [Ekspresi kueri pencarian vektor](#)
- [Perintah INFO](#)
- [Keamanan pencarian vektor](#)

Indeks dan ruang kunci Redis OSS

Indeks dibangun dan dipelihara di atas subset dari keyspace OSS Redis. Beberapa indeks dapat memilih subset yang terpisah atau tumpang tindih dari keyspace Redis tanpa batasan. OSS Ruang kunci untuk setiap indeks ditentukan oleh daftar awalan kunci yang disediakan saat indeks dibuat. Daftar awalan adalah opsional dan jika dihilangkan, seluruh OSS ruang kunci Redis akan menjadi bagian dari indeks itu. Indeks juga diketik karena hanya mencakup kunci yang memiliki tipe yang cocok. Saat ini, hanya JSON dan HASH indeks yang didukung. HASHIndeks hanya mengindeks HASH kunci yang dicakup oleh daftar awalannya dan juga JSON indeks hanya mengindeks JSON kunci yang dicakup oleh daftar awalannya. Kunci dalam daftar awalan ruang kunci indeks yang tidak memiliki tipe yang ditentukan diabaikan dan tidak memengaruhi operasi pencarian.

Ketika JSON perintah HASH atau memodifikasi kunci yang berada dalam ruang kunci indeks indeks yang diperbarui. Proses ini melibatkan penggalian bidang yang dideklarasikan untuk setiap indeks dan memperbarui indeks dengan nilai baru. Proses pembaruan dilakukan di atas latar belakang, yang berarti bahwa indeks pada akhirnya hanya konsisten dengan konten keyspace mereka. Dengan demikian sisipan atau pembaruan kunci tidak akan terlihat di hasil pencarian untuk waktu yang

singkat. Selama periode beban sistem yang berat dan/atau mutasi data yang berat, penundaan visibilitas bisa menjadi lebih lama.

Pembuatan indeks adalah proses multi-langkah. Langkah pertama adalah mengeksekusi [FT.CREATE](#) perintah yang mendefinisikan indeks. Eksekusi yang berhasil membuat secara otomatis memulai langkah kedua - penimbunan ulang. Proses pengisian ulang berjalan di thread latar belakang dan memindai ruang OSS kunci Redis mencari kunci yang ada dalam daftar awalan indeks baru. Setiap kunci yang ditemukan ditambahkan ke indeks. Akhirnya seluruh keyspace dipindai, menyelesaikan proses pembuatan indeks. Perhatikan bahwa saat proses pengisian ulang berjalan, mutasi kunci yang diindeks diizinkan, tidak ada batasan dan proses pengisian ulang indeks tidak akan selesai sampai semua kunci diindeks dengan benar. Operasi kueri yang dicoba saat indeks sedang menjalani pengisian ulang tidak diizinkan dan diakhiri dengan kesalahan. Penyelesaian proses backfilling dapat ditentukan dari output FT.INFO perintah untuk indeks tersebut ('backfill_status').

Jenis bidang indeks

Setiap bidang (kolom) indeks memiliki jenis tertentu yang dideklarasikan saat indeks dibuat dan lokasi di dalam kunci. Untuk HASH kunci lokasi adalah nama bidang di dalam HASH. Untuk JSON kunci lokasi adalah deskripsi JSON jalur. Ketika kunci dimodifikasi, data yang terkait dengan bidang yang dideklarasikan diekstraksi, dikonversi ke tipe yang dideklarasikan dan disimpan dalam indeks. Jika data hilang atau tidak berhasil dikonversi ke tipe yang dideklarasikan, maka bidang tersebut dihilangkan dari indeks. Ada empat jenis bidang, seperti yang dijelaskan sebagai berikut:

- Bidang angka berisi satu nomor. Untuk JSON bidang, aturan numerik JSON angka harus diikuti. Untuk HASH, bidang ini diharapkan berisi ASCII teks angka yang ditulis dalam format standar untuk nomor tetap atau floating point. Terlepas dari representasi dalam kunci, bidang ini dikonversi ke nomor floating point 64-bit untuk penyimpanan dalam indeks. Bidang angka dapat digunakan dengan operator pencarian rentang. Karena angka yang mendasarinya disimpan di floating point dengan batasan presisi, aturan umum tentang perbandingan numerik untuk nomor floating point berlaku.
- Bidang tag berisi nol atau lebih nilai tag yang dikodekan sebagai string UTF-8 tunggal. String diuraikan menjadi nilai tag menggunakan karakter pemisah (default adalah koma tetapi dapat diganti) dengan spasi putih depan dan belakang dihapus. Sejumlah nilai tag dapat terkandung dalam satu bidang tag. Bidang tag dapat digunakan untuk memfilter kueri untuk kesetaraan nilai tag dengan perbandingan case-sensitive atau case-insensitive.

- Bidang teks berisi gumpalan byte yang tidak perlu sesuai UTF -8. Bidang teks dapat digunakan untuk menghias hasil kueri dengan nilai-nilai yang bermakna aplikasi. Misalnya a URL atau isi dokumen, dll.
- Bidang vektor berisi vektor angka yang juga dikenal sebagai embedding. Bidang vektor mendukung K-pencarian tetangga terdekat (KNN) dari vektor berukuran tetap menggunakan algoritma dan metrik jarak tertentu. Untuk HASH indeks, bidang harus berisi seluruh vektor yang dikodekan dalam format biner (IEEElittle-endian 754). Untuk JSON kunci, jalur harus mereferensikan array dengan ukuran yang benar yang diisi dengan angka. Perhatikan bahwa ketika JSON array digunakan sebagai bidang vektor, representasi internal array dalam JSON kunci diubah menjadi format yang diperlukan oleh algoritma yang dipilih, mengurangi konsumsi memori dan presisi. Operasi baca selanjutnya menggunakan JSON perintah akan menghasilkan nilai presisi yang berkurang.

Algoritma indeks vektor

Dua algoritma indeks vektor disediakan:

- Flat — Algoritma Flat adalah pemrosesan linier brute force dari setiap vektor dalam indeks, menghasilkan jawaban yang tepat dalam batas-batas ketepatan perhitungan jarak. Karena pemrosesan linier indeks, waktu berjalan untuk algoritma ini bisa sangat tinggi untuk indeks besar.
- HNSW(Hierarchical Navigable Small Worlds) — HNSW Algoritma adalah alternatif yang memberikan perkiraan jawaban yang benar dengan imbalan waktu eksekusi yang jauh lebih rendah. Algoritma dikendalikan oleh tiga parameter `M`, `EF_CONSTRUCTION` dan `EF_RUNTIME`. Dua parameter pertama ditentukan pada waktu pembuatan indeks dan tidak dapat diubah. `EF_RUNTIME` Parameter memiliki nilai default yang ditentukan pada pembuatan indeks, tetapi dapat diganti pada setiap operasi kueri individu sesudahnya. Ketiga parameter ini berinteraksi untuk menyeimbangkan memori dan CPU konsumsi selama operasi konsumsi dan kueri serta mengontrol kualitas perkiraan KNN pencarian yang tepat (dikenal sebagai rasio penarikan).

Kedua algoritma pencarian vektor (Flat dan HNSW) mendukung `INITIAL_CAP` parameter opsional. Ketika ditentukan, parameter ini mengalokasikan memori untuk indeks, sehingga mengurangi overhead manajemen memori dan peningkatan tingkat konsumsi vektor.

Algoritma pencarian vektor seperti HNSW mungkin tidak efisien menangani penghapusan atau penipaan vektor yang dimasukkan sebelumnya. Penggunaan operasi ini dapat mengakibatkan

konsumsi memori indeks berlebih dan/atau penurunan kualitas penarikan. Pengindeksan ulang adalah salah satu metode untuk memulihkan penggunaan dan/atau ingatan memori yang optimal.

Ekspresi kueri pencarian vektor

[FT. SEARCH](#) dan [FT. AGGREGATE](#) perintah membutuhkan ekspresi kueri. Ekspresi ini adalah parameter string tunggal yang terdiri dari satu atau lebih operator. Setiap operator menggunakan satu bidang dalam indeks untuk mengidentifikasi subset kunci dalam indeks. Beberapa operator dapat digabungkan menggunakan penggabung boolean serta tanda kurung untuk lebih meningkatkan atau membatasi kumpulan kunci yang dikumpulkan (atau set hasil).

Wildcard

Operator wildcard, tanda bintang (*), cocok dengan semua kunci dalam indeks.

Rentang numerik

Operator rentang numerik memiliki sintaks berikut:

```
<range-search> ::= '@' <numeric-field-name> ':' '[' <bound> <bound> ']'  
<bound> ::= <number> | '(' <number>  
<number> ::= <integer> | <fixed-point> | <floating-point> | 'Inf' | '-Inf' | '+Inf'
```

< numeric-field-name > harus berupa bidang tipe yang dideklarasikan NUMERIC. Secara default terikat bersifat inklusif tetapi tanda kurung terbuka terkemuka '[' ('] dapat digunakan untuk membuat eksklusif terikat. Pencarian rentang dapat dikonversi menjadi perbandingan relasional tunggal (<, <=, >, >=) dengan menggunakan Inf, +Inf atau -Inf sebagai salah satu batas. Terlepas dari format numerik yang ditentukan (integer, fixed-point, floating-point, infinity) jumlahnya diubah menjadi floating point 64-bit untuk melakukan perbandingan, mengurangi presisi yang sesuai.

Example Contoh

```
@numeric-field:[0 10] // 0 <= <value> <= 10  
@numeric-field:[(0 10] // 0 < <value> <= 10  
@numeric-field:[0 (10] // 0 <= <value> < 10  
@numeric-field:[(0 (10] // 0 < <value> < 10  
@numeric-field:[1.5 (Inf] // 1.5 <= value
```

Bandingkan tag

Operator perbandingan tag memiliki sintaks berikut:

```
<tag-search> ::= '@' <tag-field-name> ':' '{' <tag> [ '|' <tag> ]* '}'
```

Jika salah satu tag di operator cocok dengan salah satu tag di bidang tag catatan, maka catatan tersebut disertakan dalam kumpulan hasil. Bidang yang dirancang oleh <tag-field-name> harus berupa bidang indeks yang dideklarasikan dengan tipe TAG. Contoh perbandingan tag adalah:

```
@tag-field:{ atag }
@tag-field: { tag1 | tag2 }
```

Kombinasi Boolean

Set hasil dari operator numerik atau tag dapat digabungkan menggunakan logika boolean: dan/atau. Tanda kurung dapat digunakan untuk mengelompokkan operator dan/atau mengubah urutan evaluasi. Sintaks operator logika boolean adalah:

```
<expression> ::= <phrase> | <phrase> '|' <expression> | '(' <expression> ')'
<phrase> ::= <term> | <term> <phrase>
<term> ::= <range-search> | <tag-search> | '*'
```

Beberapa istilah yang digabungkan menjadi frasa adalah “dan” -ed. Beberapa frasa yang dikombinasikan dengan pipa (|) adalah “atau” -ed.

Pencarian vektor

Indeks vektor mendukung dua metode pencarian yang berbeda: tetangga terdekat dan jangkauan. Pencarian tetangga terdekat menemukan angka, K, dari vektor dalam indeks yang paling dekat dengan vektor (referensi) yang disediakan — ini dalam bahasa sehari-hari disebut sebagai KNN tetangga terdekat 'K'. Sintaks untuk KNN pencarian adalah:

```
<vector-knn-search> ::= <expression> '=>[KNN' <k> '@' <vector-field-name> '$'
  <parameter-name> <modifiers> ']'
<modifiers> ::= [ 'EF_RUNTIME' <integer> ] [ 'AS' <distance-field-name>]
```

KNN Pencarian vektor hanya diterapkan pada vektor yang memenuhi yang dapat berupa kombinasi dari operator <expression> yang didefinisikan di atas: wildcard, pencarian rentang, pencarian tag dan/atau kombinasi boolean daripadanya.

- <k> adalah bilangan bulat yang menentukan jumlah vektor tetangga terdekat yang akan dikembalikan.

- `<vector-field-name>` harus menentukan bidang tipe yang dideklarasikan VECTOR.
- `<parameter-name>` bidang menentukan salah satu entri untuk PARAM tabel perintah FT . SEARCH atau FT . AGGREGATE. Parameter ini adalah nilai vektor referensi untuk perhitungan jarak. Nilai vektor dikodekan ke dalam PARAM nilai dalam format biner IEEE754 endian kecil (sama dikodekan seperti untuk bidang vektor) HASH
- Untuk jenis indeks vektor HNSW, EF_RUNTIME klausa opsional dapat digunakan untuk mengganti nilai default EF_RUNTIME parameter yang ditetapkan saat indeks dibuat.
- Opsional `<distance-field-name>` memberikan nama bidang untuk kumpulan hasil yang berisi jarak yang dihitung antara vektor referensi dan kunci yang terletak.

Pencarian rentang menempatkan semua vektor dalam jarak tertentu (radius) dari vektor referensi.

Sintaks untuk pencarian rentang adalah:

```
<vector-range-search> ::= '@' <vector-field-name> ':' '[' 'VECTOR_RANGE' ( <radius> |
'$' <radius-parameter> ) $<reference-vector-parameter> ']' [ '=' '>' '{' <modifiers>
'}' ]
<modifiers> ::= <modifier> | <modifiers>, <modifier>
<modifier> ::= [ '$yield_distance_as' ':' <distance-field-name> ] [ '$epsilon' ':'
<epsilon-value> ]
```

Di mana:

- `<vector-field-name>` adalah nama bidang vektor yang akan dicari.
- `<radius>` or `$<radius-parameter>` adalah batas jarak numerik untuk pencarian.
- `$<reference-vector-parameter>` adalah nama parameter yang berisi vektor referensi. Nilai vektor dikodekan ke dalam PARAM nilai dalam format biner IEEE 754 endian kecil (pengkodean yang sama seperti untuk bidang vektor) HASH
- Opsional `<distance-field-name>` memberikan nama bidang untuk kumpulan hasil yang berisi jarak yang dihitung antara vektor referensi dan setiap tombol.
- Opsional `<epsilon-value>` mengontrol batas operasi pencarian, vektor dalam jarak dilalui `<radius> * (1.0 + <epsilon-value>)` mencari hasil kandidat. Defaultnya adalah .01.

Perintah INFO

Pencarian vektor menambah OSS [INFO](#) perintah Redis dengan beberapa bagian statistik dan penghitung tambahan. Permintaan untuk mengambil bagian SEARCH akan mengambil semua bagian berikut:

Bagian `search_memory`

Nama	Penjelasan
<code>search_used_memory_bytes</code>	Jumlah byte memori yang dikonsumsi di semua struktur data pencarian
<code>search_used_memory_human</code>	Versi yang dapat dibaca manusia di atas

Bagian `search_index_stats`

Nama	Penjelasan
<code>search_number_of_indexes</code>	Jumlah indeks yang dibuat
<code>search_num_fulltext_indexes</code>	Jumlah bidang non-vektor di semua indeks
<code>search_num_vector_indexes</code>	Jumlah bidang vektor di semua indeks
<code>search_num_hash_indexes</code>	Jumlah indeks pada tombol HASH tipe
<code>search_num_json_indexes</code>	Jumlah indeks pada tombol JSON tipe
<code>search_total_indexed_keys</code>	Jumlah total kunci di semua indeks
<code>search_total_indexed_vectors</code>	Jumlah total vektor di semua indeks
<code>search_total_indexed_hash_keys</code>	Jumlah total kunci tipe HASH di semua indeks
<code>search_total_indexed_json_keys</code>	Jumlah total kunci tytpe JSON di semua indeks
<code>search_total_index_size</code>	Byte yang digunakan oleh semua indeks

Nama	Penjelasan
search_total_fulltext_index_size	Byte yang digunakan oleh struktur indeks non-vektor
search_total_vector_index_size	Byte yang digunakan oleh struktur indeks vektor
search_max_index_lag_ms	Penundaan konsumsi selama pembaruan batch konsumsi terakhir

Bagian **search_ingestion**

Nama	Penjelasan
search_background_indexing_status	Status konsumsi. NO_ACTIVITY berarti mengganggu. Nilai lain menunjukkan ada kunci dalam proses dicerna.
search_ingestion_dijeda	Kecuali saat memulai ulang, ini harus selalu "tidak".

Bagian **search_backfill**

Note

Beberapa bidang yang didokumentasikan di bagian ini hanya terlihat saat pengisian ulang sedang berlangsung.

Nama	Penjelasan
search_num_active_backfills	Jumlah kegiatan pengurukan saat ini
search_backfills_dijeda	Kecuali ketika kehabisan ingatan, ini harus selalu "tidak".

Nama	Penjelasan
search_current_backfill_progress_percentage	% penyelesaian (0-100) dari isi ulang saat ini

Bagian `search_query`

Nama	Penjelasan
search_num_active_queries	Jumlah <code>FT.SEARCH</code> dan <code>FT.AGGREGATE</code> perintah yang sedang berlangsung

Keamanan pencarian vektor

[Redis OSS ACL \(Access Control Lists\)](#) mekanisme keamanan untuk kedua perintah dan akses data diperluas untuk mengontrol fasilitas pencarian. ACL kontrol perintah pencarian individu sepenuhnya didukung. ACL kategori baru, `@search`, disediakan dan banyak kategori yang ada (`@fast`, `@read`, `@write`, dll.) Diperbarui untuk menyertakan perintah baru. Perintah pencarian tidak memodifikasi data kunci, artinya ACL mesin yang ada untuk akses tulis dipertahankan. Aturan akses untuk HASH dan JSON operasi tidak dimodifikasi oleh kehadiran indeks; kontrol akses tingkat kunci normal masih diterapkan pada perintah tersebut.

Perintah pencarian dengan indeks juga memiliki akses mereka dikendalikan melalui Redis OSSACL. Pemeriksaan akses dilakukan pada tingkat seluruh indeks, bukan pada tingkat per-kunci. Ini berarti bahwa akses ke indeks diberikan kepada pengguna hanya jika pengguna tersebut memiliki izin untuk mengakses semua kunci yang mungkin dalam daftar awalan keyspace indeks tersebut. Dengan kata lain, konten sebenarnya dari indeks tidak mengontrol akses. Sebaliknya, itu adalah isi teoritis dari indeks seperti yang didefinisikan oleh daftar awalan yang digunakan untuk pemeriksaan keamanan. Hal ini dapat mudah untuk membuat situasi di mana pengguna telah membaca dan/atau menulis akses ke kunci tetapi tidak dapat mengakses indeks yang berisi kunci itu. Perhatikan bahwa hanya akses baca ke ruang kunci yang diperlukan untuk membuat atau menggunakan indeks - ada atau tidak adanya akses tulis tidak dipertimbangkan.

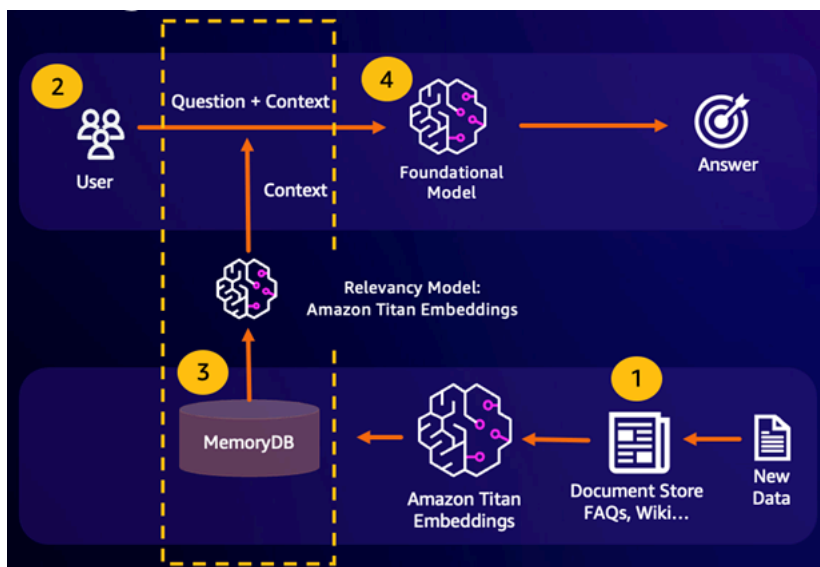
Untuk informasi selengkapnya tentang penggunaan ACLs dengan MemoryDB, lihat [Mengautentikasi pengguna dengan Daftar Kontrol Akses](#) (). ACLs

Kasus penggunaan

Berikut ini adalah kasus penggunaan pencarian vektor.

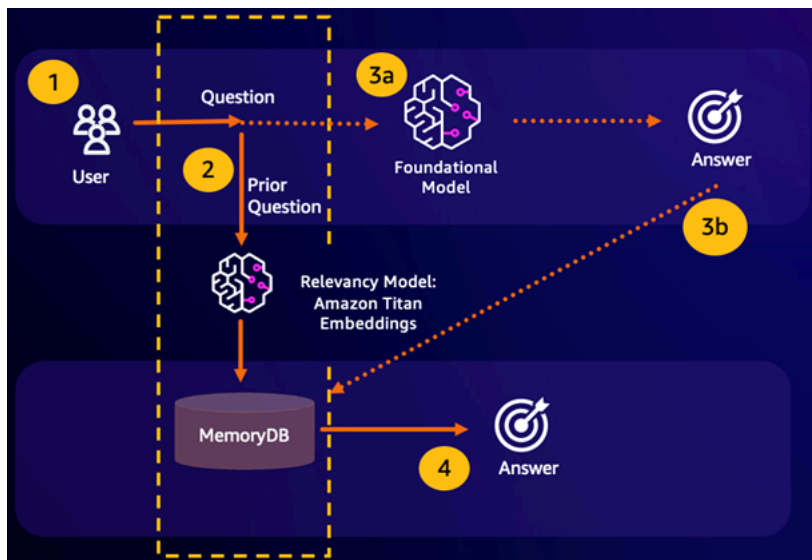
Pengambilan Generasi Augmented () RAG

Retrieval Augmented Generation (RAG) memanfaatkan pencarian vektor untuk mengambil bagian yang relevan dari kumpulan data yang besar untuk menambah model bahasa besar (). LLM Secara khusus, encoder menyematkan konteks input dan kueri penelusuran ke dalam vektor, kemudian menggunakan perkiraan pencarian tetangga terdekat untuk menemukan bagian yang serupa secara semantik. Bagian yang diambil ini digabungkan dengan konteks asli untuk memberikan informasi tambahan yang relevan untuk mengembalikan respons yang lebih akurat LLM kepada pengguna.



Cache Semantik Tahan Lama

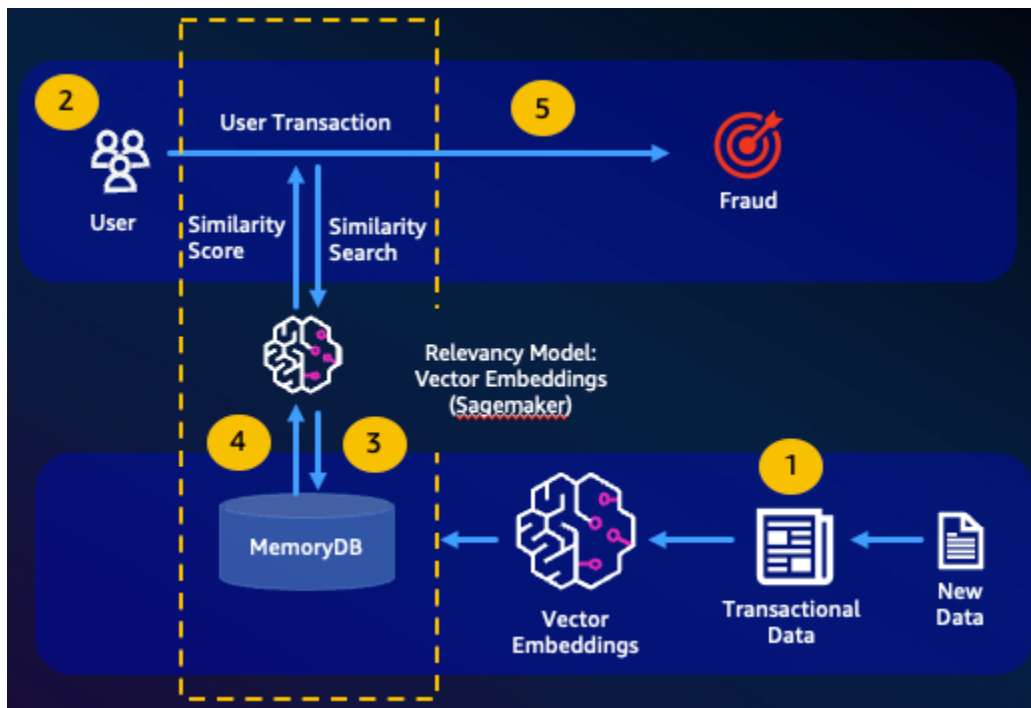
Semantic Caching adalah proses untuk mengurangi biaya komputasi dengan menyimpan hasil sebelumnya dari FM. Dengan menggunakan kembali hasil sebelumnya dari kesimpulan sebelumnya alih-alih mengkomputernya kembali, caching semantik mengurangi jumlah perhitungan yang diperlukan selama inferensi melalui. FMs MemoryDB memungkinkan caching semantik yang tahan lama, yang menghindari hilangnya data dari kesimpulan masa lalu Anda. Ini memungkinkan aplikasi AI generatif Anda merespons dalam milidetik satu digit dengan jawaban dari pertanyaan serupa secara semantik sebelumnya, sekaligus mengurangi biaya dengan menghindari kesimpulan yang tidak perlu. LLM



- **Pencarian semantik** — Jika kueri pelanggan semantik serupa berdasarkan skor kesamaan yang ditentukan dengan pertanyaan sebelumnya, memori buffer FM (MemoryDB) akan mengembalikan jawaban ke pertanyaan sebelumnya di langkah 4 dan tidak akan memanggil FM melalui langkah 3. Ini akan menghindari latensi model pondasi (FM) dan biaya yang dikeluarkan, memberikan pengalaman yang lebih cepat bagi pelanggan.
- **Kehilangan pencarian semantik** — Jika kueri pelanggan tidak semantik serupa berdasarkan skor kesamaan yang ditentukan dengan kueri sebelumnya, pelanggan akan menghubungi FM untuk memberikan respons kepada pelanggan pada langkah 3a. Respon yang dihasilkan dari FM kemudian akan disimpan sebagai vektor ke dalam MemoryDB untuk kueri future (langkah 3b) untuk meminimalkan biaya FM pada pertanyaan semantik serupa. Dalam alur ini, langkah 4 tidak akan dipanggil karena tidak ada pertanyaan semantik serupa untuk kueri asli.

Deteksi penipuan

Deteksi penipuan, suatu bentuk deteksi anomali, mewakili transaksi yang valid sebagai vektor sambil membandingkan representasi vektor dari transaksi baru bersih. Penipuan terdeteksi ketika transaksi baru bersih ini memiliki kesamaan rendah dengan vektor yang mewakili data transaksional yang valid. Hal ini memungkinkan penipuan dideteksi dengan memodelkan perilaku normal, daripada mencoba memprediksi setiap kemungkinan kejadian penipuan. MemoryDB memungkinkan organisasi untuk melakukan ini dalam periode throughput tinggi, dengan positif palsu minimal dan latensi milidetik satu digit.



Kasus penggunaan lainnya

- Mesin rekomendasi dapat menemukan pengguna produk atau konten serupa dengan mewakili item sebagai vektor. Vektor dibuat dengan menganalisis atribut dan pola. Berdasarkan pola dan atribut pengguna, item baru yang tidak terlihat dapat direkomendasikan kepada pengguna dengan menemukan vektor paling mirip yang telah dinilai secara positif selaras dengan pengguna.
- Mesin pencari dokumen mewakili dokumen teks sebagai vektor angka yang padat, menangkap makna semantik. Pada waktu pencarian, mesin mengubah permintaan pencarian ke vektor dan menemukan dokumen dengan vektor yang paling mirip dengan kueri menggunakan perkiraan pencarian tetangga terdekat. Pendekatan kesamaan vektor ini memungkinkan pencocokan dokumen berdasarkan makna daripada hanya mencocokkan kata kunci.

Fitur pencarian vektor dan batas

Ketersediaan pencarian vektor

Konfigurasi MemoryDB yang diaktifkan pencarian vektor didukung pada tipe node R6g, R7g, dan T4G dan tersedia di semua Wilayah di mana MemoryDB tersedia. AWS

Cluster yang ada tidak dapat dimodifikasi untuk mengaktifkan pencarian. Namun, cluster yang mendukung pencarian dapat dibuat dari snapshot cluster dengan penelusuran dinonaktifkan.

Pembatasan parametrik

Tabel berikut menunjukkan batas untuk berbagai item pencarian vektor:

Item	Nilai maksimum
Jumlah dimensi dalam vektor	32768
Jumlah indeks yang dapat dibuat	10
Jumlah bidang dalam indeks	50
KAKI. SEARCH dan FT. AGGREGATE TIMEOUT klausula (milidetik)	10000
Jumlah tahapan pipa di FT. AGGREGATE perintah	32
Jumlah bidang di FT. AGGREGATE LOAD klausula	1024
Jumlah bidang di FT. AGGREGATE GROUP BY klausula	16
Jumlah bidang di FT. AGGREGATE SORT BY klausula	16
Jumlah parameter di FT. AGGREGATE PARAM klausula	32
HNSW Parameter M	512
HNSW Parameter EF_CONSTRUCTION	4096
HNSW Parameter EF_RUNTIME	4096

Batas penskalaan

Pencarian vektor untuk MemoryDB saat ini terbatas pada pecahan tunggal dan penskalaan horizontal tidak didukung. Pencarian vektor mendukung penskalaan vertikal dan replika.

Pembatasan operasional

Indeks Persistensi dan Penimbunan Ulang

Fitur pencarian vektor mempertahankan definisi indeks, dan isi indeks. Ini berarti bahwa selama setiap permintaan operasional atau peristiwa yang menyebabkan node memulai atau memulai ulang, definisi indeks dan konten dipulihkan dari snapshot terbaru dan setiap transaksi yang tertunda diputar ulang dari Jurnal. Tidak ada tindakan pengguna yang diperlukan untuk memulai ini. Pembangunan kembali dilakukan sebagai operasi pengurukan segera setelah data dipulihkan. [Ini secara fungsional setara dengan sistem yang secara otomatis mengeksekusi FT. CREATE](#) perintah untuk setiap indeks yang ditentukan. Perhatikan bahwa node menjadi tersedia untuk operasi aplikasi segera setelah data dipulihkan tetapi kemungkinan sebelum pengurukan indeks selesai, yang berarti bahwa isi ulang akan kembali terlihat oleh aplikasi, misalnya, perintah pencarian menggunakan indeks backfilling dapat ditolak. Untuk informasi lebih lanjut tentang penimbunan ulang, lihat [Ikhtisar pencarian vektor](#).

Penyelesaian isi ulang indeks tidak disinkronkan antara primer dan replika. Kurangnya sinkronisasi ini secara tak terduga dapat terlihat oleh aplikasi dan oleh karena itu disarankan agar aplikasi memverifikasi penyelesaian pengisian ulang pada primer dan semua replika sebelum memulai operasi pencarian.

Impor/ekspor snapshot dan Migrasi Langsung

Kehadiran indeks pencarian dalam RDB file membatasi transportabilitas yang kompatibel dari data tersebut. Format indeks vektor yang ditentukan oleh fungsionalitas pencarian vektor MemoryDB hanya dipahami oleh cluster yang diaktifkan vektor MemoryDB lainnya. Selain itu, RDB file dari kluster pratinjau dapat diimpor oleh kluster MemoryDB versi GA, yang akan membangun kembali konten indeks saat memuat file. RDB

Namun, RDB file yang tidak mengandung indeks tidak dibatasi dengan cara ini. Dengan demikian data dalam kluster pratinjau dapat diekspor ke kluster non-pratinjau dengan menghapus indeks sebelum ekspor.

Konsumsi memori

Konsumsi memori didasarkan pada jumlah vektor, jumlah dimensi, nilai-M, dan jumlah data non-vektor, seperti metadata yang terkait dengan vektor atau data lain yang disimpan dalam instance.

Total memori yang dibutuhkan adalah kombinasi dari ruang yang dibutuhkan untuk data vektor aktual, dan ruang yang diperlukan untuk indeks vektor. Ruang yang diperlukan untuk data Vektor

dihitung dengan mengukur kapasitas aktual yang diperlukan untuk menyimpan vektor di dalam HASH atau struktur JSON data dan overhead ke pelat memori terdekat, untuk alokasi memori yang optimal. Setiap indeks vektor menggunakan referensi ke data vektor yang disimpan dalam struktur data ini, dan menggunakan pengoptimalan memori yang efisien untuk menghapus salinan duplikat data vektor dalam indeks.

Jumlah vektor tergantung pada bagaimana Anda memutuskan untuk mewakili data Anda sebagai vektor. Misalnya, Anda dapat memilih untuk mewakili satu dokumen menjadi beberapa potongan, di mana setiap potongan mewakili vektor. Atau, Anda dapat memilih untuk mewakili seluruh dokumen sebagai vektor tunggal.

Jumlah dimensi vektor Anda tergantung pada model penyematan yang Anda pilih. Misalnya, jika Anda memilih untuk menggunakan model embedding [AWS Titan](#) maka jumlah dimensi akan menjadi 1536.

Parameter M mewakili jumlah tautan dua arah yang dibuat untuk setiap elemen baru selama konstruksi indeks. MemoryDB default nilai ini ke 16; Namun, Anda dapat mengganti ini. Parameter M yang lebih tinggi bekerja lebih baik untuk dimensi tinggi dan/atau persyaratan penarikan tinggi sementara parameter M rendah bekerja lebih baik untuk dimensi rendah dan/atau persyaratan penarikan rendah. Nilai M meningkatkan konsumsi memori karena indeks menjadi lebih besar, meningkatkan konsumsi memori.

Dalam pengalaman konsol, MemoryDB menawarkan cara mudah untuk memilih jenis instans yang tepat berdasarkan karakteristik beban kerja vektor Anda setelah memeriksa Aktifkan pencarian vektor di bawah pengaturan cluster.

Cluster settings

Enable vector search [Info](#)

You can store vector embeddings and perform vector similarity searches.

i Vector search is compatible with MemoryDB version 7.1 in a single shard configuration. Once the cluster is created with vector search enabled, the number of shards cannot be modified.

Redis version compatibility

Version compatibility of the Redis engine that will run on your nodes.



Port

The port number that nodes accept connections on.

Parameter groups

Parameter groups control the runtime properties of your nodes and clusters.



Node type

The type of node to be deployed and its associated memory size.

13.07 GiB memory Up to 12.5 Gigabit network performance

[Use vector calculator](#)

Number of shards

Enter the number of shards, from 1 to 500.

Replica nodes per shard

Enter the number of replica nodes for each shard, from 0 to 5.

Contoh beban kerja

Seorang pelanggan ingin membangun mesin pencari semantik yang dibangun di atas dokumen keuangan internal mereka. Mereka saat ini memegang 1M dokumen keuangan yang dipotong menjadi 10 vektor per dokumen menggunakan model embedding titan dengan 1536 dimensi dan tidak memiliki data non-vektor. Pelanggan memutuskan untuk menggunakan default 16 sebagai parameter M.

- Vektor: $1 \text{ M} * 10 \text{ potongan} = 10\text{M}$ vektor
- Dimensi: 1536
- Data Non-Vektor (GB): 0 GB
- Parameter M: 16

Dengan data ini, pelanggan dapat mengklik tombol Gunakan kalkulator vektor di dalam konsol untuk mendapatkan jenis instans yang direkomendasikan berdasarkan parameternya:

Vector calculator



Vector calculator will use your inputs to provide you with an estimate for your node type. [Learn more](#)

Number of vectors

Number of dimensions

Dimensionality of vectors

Amount of non-vector data (GiB) - optional

Estimated amount of metadata and other non-vector data

M parameter - optional

M parameter represents the number of bi-directional links created for every new element during construction

A reasonable range for M is 2-512. Higher M parameters work better on datasets with high dimensionality and/or high recall, while lower M parameters work better for datasets with low dimensionality and/or low recalls. The default M parameter is 16.

Cancel

Calculate

Node type

The type of node to be deployed and its associated memory size.

db.r7g.4xlarge

105.81 GiB memory Up to 15 Gigabit network performance

Use vector calculator

The recommended node type is based on your input to the vector calculator.

Dalam contoh ini, kalkulator vektor akan mencari [jenis node r7g MemoryDB](#) terkecil yang dapat menampung memori yang diperlukan untuk menyimpan vektor berdasarkan parameter yang disediakan. Perhatikan bahwa ini adalah perkiraan, dan Anda harus menguji jenis instance untuk memastikannya sesuai dengan kebutuhan Anda.

Berdasarkan metode perhitungan di atas dan parameter dalam beban kerja sampel, data vektor ini akan membutuhkan 104,9 GB untuk menyimpan data dan indeks tunggal. Dalam hal ini, jenis `db.r7g.4xlarge` instance akan direkomendasikan karena memiliki 105,81 GB memori yang dapat digunakan. Tipe node terkecil berikutnya akan terlalu kecil untuk menahan beban kerja vektor.

Karena masing-masing indeks vektor menggunakan referensi ke data vektor yang disimpan dan tidak membuat salinan tambahan dari data vektor dalam indeks vektor, indeks juga akan mengkonsumsi ruang yang relatif lebih sedikit. Ini sangat berguna dalam membuat beberapa indeks, dan juga dalam situasi di mana bagian dari data vektor telah dihapus dan merekonstruksi HNSW grafik akan membantu menciptakan koneksi simpul yang optimal untuk hasil pencarian vektor berkualitas tinggi.

Keluar dari Memori saat mengisi ulang

Mirip dengan operasi OSS tulis Redis, isi ulang indeks dikenakan batasan. out-of-memory Jika OSS memori Redis terisi saat pengisian ulang sedang berlangsung, semua isi ulang dijeda. Jika memori tersedia, proses pengurukan dilanjutkan. Dimungkinkan juga untuk menghapus dan mengindeks saat pengisian ulang dijeda karena kehabisan memori.

Transaksi

Perintah `FT.CREATE`, `FT.DROPINDEX`, `FT.ALIASADD`, `FT.ALIASDEL`, dan `FT.ALIASUPDATE` tidak dapat dieksekusi dalam konteks transaksional, yaitu, tidak di dalam `EXEC` blok `MULTI`/atau dalam skrip `LUA` atau `FUNCTION`.

Menggunakan AWS Management Console

Untuk membuat cluster yang diaktifkan untuk pencarian vektor di dalam konsol, Anda perlu mengaktifkan pencarian vektor di bawah pengaturan Cluster. Pencarian vektor tersedia untuk MemoryDB versi 7.2 dalam konfigurasi pecahan tunggal.

Cluster settings

Enable vector search [Info](#)

You can store vector embeddings and perform vector similarity searches.

i Vector search is compatible with MemoryDB version 7.1 in a single shard configuration. Once the cluster is created with vector search enabled, the number of shards cannot be modified.

Untuk informasi lebih lanjut tentang menggunakan pencarian vektor dengan AWS Management Console, lihat [Membuat cluster \(Konsol\)](#).

Menggunakan AWS Command Line Interface

Untuk membuat kluster MemoryDB yang diaktifkan pencarian vektor, Anda dapat menggunakan perintah [create-cluster](#) MemoryDB dengan meneruskan grup parameter yang tidak dapat diubah untuk mengaktifkan kemampuan pencarian vektor. `default.memorydb-redis7.search`

```
aws memorydb create-cluster \  
  --cluster-name <value> \  
  --node-type <value> \  
  --engine redis \  
  --engine-version 7.1 \  
  --num-shards 1 \  
  --acl-name <value> \  
  --parameter-group-name default.memorydb-redis7.search
```

Secara opsional, Anda juga dapat membuat grup parameter baru untuk mengaktifkan pencarian vektor seperti yang ditunjukkan pada contoh berikut. Anda dapat mempelajari lebih lanjut tentang grup parameter [di sini](#).

```
aws memorydb create-parameter-group \  
  --parameter-group-name my-search-parameter-group \  
  --family memorydb_redis7
```

Selanjutnya, perbarui parameter yang diaktifkan pencarian ke ya di grup parameter yang baru dibuat.

```
aws memorydb update-parameter-group \  
  --parameter-group-name my-search-parameter-group \  
  --parameter-name-values "ParameterName=search-enabled,ParameterValue=yes"
```

Anda sekarang dapat menggunakan grup parameter kustom ini alih-alih grup parameter default untuk mengaktifkan pencarian vektor pada cluster MemoryDB Anda.

Perintah pencarian vektor

Berikut ini adalah daftar perintah yang didukung untuk pencarian vektor.

Topik

- [KAKI.CREATE](#)

- [KAKI. SEARCH](#)
- [KAKI. AGGREGATE](#)
- [KAKI. DROPINDEX](#)
- [KAKI. INFO](#)
- [KAKI. _ LIST](#)
- [KAKI. ALIASADD](#)
- [KAKI. ALIASDEL](#)
- [KAKI. ALIASUPDATE](#)
- [KAKI. _ ALIASLIST](#)
- [KAKI. PROFILE](#)
- [KAKI. EXPLAIN](#)
- [KAKI. EXPLAINCLI](#)

KAKI. CREATE

Membuat indeks dan memulai pengurutan indeks itu. Untuk informasi selengkapnya, lihat [Ikhtisar pencarian vektor](#) untuk detail tentang konstruksi indeks.

Sintaksis

```
FT.CREATE <index-name>
ON HASH | JSON
[PREFIX <count> <prefix1> [<prefix2>...]]
SCHEMA
(<field-identifier> [AS <alias>]
  NUMERIC
  | TAG [SEPARATOR <sep>] [CASESENSITIVE]
  | TEXT
  | VECTOR [HNSW|FLAT] <attr_count> [<attribute_name> <attribute_value>])
)+
```

Skema

- Pengidentifikasi bidang:
 - Untuk kunci hash, pengenal bidang adalah nama bidang.

- Untuk JSON kunci, pengenalan bidang adalah JSON jalur A.

Untuk informasi selengkapnya, lihat [Jenis bidang indeks](#).

- Jenis bidang:
 - TAG: Untuk informasi selengkapnya, lihat [Tag](#).
 - NUMERIC: Bidang berisi angka.
 - TEXT: Bidang berisi gumpalan data.
 - VECTOR: bidang vektor yang mendukung pencarian vektor.
 - Algoritma — dapat berupa HNSW (Hierarchical Navigable Small World) atau FLAT (brute force).
 - `attr_count`— jumlah atribut yang akan diteruskan sebagai konfigurasi algoritma, ini termasuk nama dan nilai.
 - `{attribute_name} {attribute_value}`— pasangan kunci/nilai khusus algoritma yang menentukan konfigurasi indeks.

Untuk FLAT algoritma, atribut adalah:

Diperlukan:

- DIM— Jumlah dimensi dalam vektor.
- DISTANCE_ METRIC - Dapat menjadi salah satu [L2 | IP | COSINE].
- TYPE— Jenis vektor. Satu-satunya jenis yang didukung adalah FLOAT32.

Opsional:

- INITIAL_ CAP — Kapasitas vektor awal dalam indeks yang mempengaruhi ukuran alokasi memori indeks.

Untuk HNSW algoritma, atribut adalah:

Diperlukan:

- TYPE— Jenis vektor. Satu-satunya jenis yang didukung adalah FLOAT32.
- DIM— Dimensi vektor, ditentukan sebagai bilangan bulat positif. Maksimal: 32768
- DISTANCE_ METRIC - Dapat menjadi salah satu [L2 | IP | COSINE].

Opsional:

- `INITIAL_CAP` — Kapasitas vektor awal dalam indeks yang mempengaruhi ukuran alokasi memori indeks. Default ke 1024.
- `M` — Jumlah tepi keluar maksimum yang diizinkan untuk setiap node dalam grafik di setiap lapisan. pada lapisan nol jumlah maksimal tepi keluar adalah 2M. Defaultnya adalah 16 Maksimum adalah 512.
- `EF_CONSTRUCTION` — mengontrol jumlah vektor yang diperiksa selama konstruksi indeks. Nilai yang lebih tinggi untuk parameter ini akan meningkatkan rasio penarikan dengan mengorbankan waktu pembuatan indeks yang lebih lama. Nilai default adalah 200. Nilai maksimum adalah 4096.
- `EF_RUNTIME` — mengontrol jumlah vektor yang diperiksa selama operasi kueri. Nilai yang lebih tinggi untuk parameter ini dapat menghasilkan penarikan yang lebih baik dengan mengorbankan waktu kueri yang lebih lama. Nilai parameter ini dapat diganti berdasarkan per-query. Nilai default adalah 10. Nilai maksimum adalah 4096.

Nilai yang ditampilkan

Mengembalikan string pesan OK sederhana atau balasan kesalahan.

Contoh

Note

Contoh berikut menggunakan argumen asli [redis-cli](#), seperti de-quoting dan de-escaping data, sebelum mengirimnya ke Redis. OSS Untuk menggunakan klien bahasa pemrograman lainnya (Python, Ruby, C #, dll.), Ikuti aturan penanganan lingkungan tersebut untuk menangani string dan data biner. Untuk informasi selengkapnya tentang klien yang didukung, lihat [Alat untuk Dibangun AWS](#)

Example 1: Buat beberapa indeks

Buat indeks untuk vektor ukuran 2

```
FT.CREATE hash_idx1 ON HASH PREFIX 1 hash: SCHEMA vec AS VEC VECTOR HNSW 6 DIM 2 TYPE
FLOAT32 DISTANCE_METRIC L2
OK
```

Buat JSON indeks 6 dimensi menggunakan HNSW algoritma:

```
FT.CREATE json_idx1 ON JSON PREFIX 1 json: SCHEMA $.vec AS VEC VECTOR HNSW 6 DIM 6 TYPE
  FLOAT32 DISTANCE_METRIC L2
OK
```

Example Contoh 2: Mengisi beberapa data

Perintah berikut diformat sehingga dapat dieksekusi sebagai argumen untuk program terminal redis-cli. Pengembang yang menggunakan klien bahasa pemrograman (seperti Python, Ruby, C #, dll.) Perlu mengikuti aturan penanganan lingkungan mereka untuk menangani string dan data biner.

Membuat beberapa data hash dan json:

```
HSET hash:0 vec "\x00\x00\x00\x00\x00\x00\x00\x00"
HSET hash:1 vec "\x00\x00\x00\x00\x00\x00\x00\x80\xbf"
JSON.SET json:0 . '{"vec":[1,2,3,4,5,6]}'
JSON.SET json:1 . '{"vec":[10,20,30,40,50,60]}'
JSON.SET json:2 . '{"vec":[1.1,1.2,1.3,1.4,1.5,1.6]}'
```

Perhatikan hal berikut:

- Kunci hash dan JSON data memiliki awalan definisi indeksinya.
- Vektor berada pada jalur yang sesuai dari definisi indeks.
- Vektor hash dimasukkan sebagai data hex sementara JSON data dimasukkan sebagai angka.
- Vektor adalah panjang yang sesuai, entri vektor hash dua dimensi memiliki data hex senilai dua float, entri vektor json enam dimensi memiliki enam angka.

Example Contoh 3: Hapus dan buat ulang indeks

```
FT.DROPINDEX json_idx1
OK

FT.CREATE json_idx1 ON JSON PREFIX 1 json: SCHEMA $.vec AS VEC VECTOR FLAT 6 DIM 6 TYPE
  FLOAT32 DISTANCE_METRIC L2
OK
```

Perhatikan JSON indeks baru menggunakan FLAT algoritma bukan HNSW algoritma. Perhatikan juga bahwa itu akan mengindeks ulang JSON data yang ada:

- **LIMIT:** <offset><count>: Klausa ini memberikan kemampuan pagination karena hanya kunci yang memenuhi nilai offset dan count yang dikembalikan. Jika klausa ini dihilangkan, defaultnya menjadi "LIMIT 10", yaitu, hanya maksimal 10 kunci yang akan dikembalikan.
- **PARAMS:** Dua kali jumlah pasangan nilai kunci. Pasangan kunci/nilai Param dapat direferensikan dari dalam ekspresi kueri. Untuk informasi selengkapnya, lihat [Ekspresi kueri penelusuran vektor](#).
- **COUNT:** Klausa ini menekan pengembalian isi kunci, hanya jumlah kunci yang dikembalikan. Ini adalah alias untuk "LIMIT 0".

Nilai yang ditampilkan

Mengembalikan array atau balasan kesalahan.

- Jika operasi selesai berhasil, ia mengembalikan array. Elemen pertama adalah jumlah total kunci yang cocok dengan query. Elemen yang tersisa adalah pasangan nama kunci dan daftar bidang. Daftar bidang adalah array lain yang terdiri dari pasangan nama bidang dan nilai.
- Jika indeks sedang dalam proses diisi kembali, perintah segera mengembalikan balasan kesalahan.
- Jika batas waktu tercapai, perintah mengembalikan balasan kesalahan.

Contoh: Lakukan beberapa pencarian

Note

Contoh berikut menggunakan argumen asli [redis-cli](#), seperti de-quoting dan de-escaping data, sebelum mengirimnya ke Redis. OSS Untuk menggunakan klien bahasa pemrograman lainnya (Python, Ruby, C #, dll.), Ikuti aturan penanganan lingkungan tersebut untuk menangani string dan data biner. Untuk informasi selengkapnya tentang klien yang didukung, lihat [Alat untuk Dibangun AWS](#)

Pencarian hash

```
FT.SEARCH hash_idx1 "*=>[KNN 2 @VEC $query_vec]" PARAMS 2 query_vec
"\x00\x00\x00\x00\x00\x00\x00\x00" DIALECT 2
1) (integer) 2
2) "hash:0"
3) 1) "__VEC_score"
```



```

4) "[{"vec\":[1.1, 1.2, 1.3, 1.4, 1.5, 1.6]}]"
4) "json:0"
5) 1) "__VEC_score"
   2) "91"
   3) "$"
   4) "[{"vec\":[1.0, 2.0, 3.0, 4.0, 5.0, 6.0]}]"
6) "json:1"
7) 1) "__VEC_score"
   2) "9100"
   3) "$"
   4) "[{"vec\":[10.0, 20.0, 30.0, 40.0, 50.0, 60.0]}]"

```

KAKI. AGGREGATE

Sebuah superset dari FT. SEARCH perintah, ini memungkinkan pemrosesan tambahan substansif dari kunci yang dipilih oleh ekspresi kueri.

Sintaksis

```

FT.AGGREGATE index query
[LOAD * | [count field [field ...]]]
[TIMEOUT timeout]
[PARAMS count name value [name value ...]]
[FILTER expression]
[LIMIT offset num]
[GROUPBY count property [property ...] [REDUCE function count arg [arg ...] [AS name]
[REDUCE function count arg [arg ...] [AS name] ...]] ...]]
[SORTBY count [ property ASC | DESC [property ASC | DESC ...]] [MAX num]]
[APPLY expression AS name]

```

- FILTER, LIMIT, GROUPBY, SORTBY dan APPLY klausa dapat diulang beberapa kali dalam urutan apa pun dan dicampur secara bebas. Mereka diterapkan dalam urutan yang ditentukan dengan output dari satu klausa yang memberi masukan dari klausa berikutnya.
- Dalam sintaks di atas, “properti” adalah bidang yang dideklarasikan di [FT. CREATE](#) perintah untuk indeks ini ATAU output dari APPLY klausa atau REDUCE fungsi sebelumnya.
- LOAD klausul dibatasi untuk memuat bidang yang telah dideklarasikan dalam indeks. “LOAD*” akan memuat semua bidang yang dideklarasikan dalam indeks.
- Fungsi peredam berikut didukung: COUNT, COUNT _DISTINCTISH, SUM, MIN, MAX, AVG, STDDEV, QUANTILE, FIRST _ TOLISTVALUE, dan RANDOM _SAMPLE. Untuk informasi selengkapnya, lihat [Agregasi](#)

- **LIMIT<offset><count>**: Mempertahankan catatan mulai <offset> dan berlanjut hingga <count>, semua catatan lainnya dibuang.
- **PARAMS**: Dua kali jumlah pasangan nilai kunci. Pasangan kunci/nilai Param dapat direferensikan dari dalam ekspresi kueri. Untuk informasi selengkapnya, lihat [Ekspresi kueri penelusuran vektor](#).

Nilai yang ditampilkan

Mengembalikan array atau balasan kesalahan.

- Jika operasi selesai berhasil, ia mengembalikan array. Elemen pertama adalah bilangan bulat tanpa arti tertentu (harus diabaikan). Elemen yang tersisa adalah hasil output pada tahap terakhir. Setiap elemen adalah array nama bidang dan pasangan nilai.
- Jika indeks sedang dalam proses diisi kembali, perintah segera mengembalikan balasan kesalahan.
- Jika batas waktu tercapai, perintah mengembalikan balasan kesalahan.

KAKI. DROPINDEX

Jatuhkan indeks. Definisi indeks dan konten terkait dihapus. OSSKunci Redis tidak terpengaruh.

Sintaksis

```
FT.DROPINDEX <index-name>
```

Nilai yang ditampilkan

Mengembalikan string pesan OK sederhana atau balasan kesalahan.

KAKI. INFO

Sintaksis

```
FT.INFO <index-name>
```

Output dari FT. INFO halaman adalah array pasangan nilai kunci seperti yang dijelaskan dalam tabel berikut:

Kunci	Tipe nilai	Deskripsi
index_name	string	Nama indeks
creation_timestamp	integer	Stempel waktu pembuatan bergaya Unix
key_type	string	HASH atau JSON
key_prefixes	array string	Awalan kunci untuk indeks ini
ladang	array informasi lapangan	Bidang indeks ini
space_usage	integer	Byte memori yang digunakan oleh indeks ini
fullext_space_usage	integer	Byte memori yang digunakan oleh bidang non-vektor
vector_space_usage	integer	Byte memori yang digunakan oleh bidang vektor
num_docs	integer	Jumlah kunci yang saat ini terdapat dalam indeks
num_indexed_vectors	integer	Jumlah vektor yang saat ini terdapat dalam indeks
current_lag	integer	Penundaan konsumsi baru-baru ini () milliseconds
backfill_status	string	Salah satu dari: Selesai, InProgress, Dijeda atau Gagal

Tabel berikut menjelaskan informasi untuk setiap bidang:

Kunci	Tipe nilai	Deskripsi
pengenal	string	nama bidang

Kunci	Tipe nilai	Deskripsi
field_name	string	Nama anggota hash atau JSON Path
jenis	string	salah satu dari: Numerik, Tag, Teks atau Vektor
pilihan	string	abaikan

Jika bidangnya bertipe Vector, informasi tambahan akan hadir tergantung pada algoritma.

Untuk HNSW algoritma:

Kunci	Tipe nilai	Deskripsi
algoritma	string	HNSW
data_type	string	FLOAT32
distance_metric	string	salah satu dari: L2, IP atau Cosine
initial_capacity	integer	Ukuran awal indeks bidang vektor
kapasitas saat ini	integer	Ukuran indeks bidang vektor saat ini
maximum_edges	integer	Parameter M saat pembuatan
ef_konstruksi	integer	CONSTRUCTIONParameter EF_ saat pembuatan
ef_runtime	integer	RUNTIMEParameter EF_ saat pembuatan

Untuk FLAT algoritma:

Kunci	Tipe nilai	Deskripsi
algoritma	string	FLAT
data_type	string	FLOAT32
distance_metric	string	salah satu dari: L2, IP atau Cosine
initial_capacity	integer	Ukuran awal indeks bidang vektor
kapasitas saat ini	integer	Ukuran indeks bidang vektor saat ini

KAKI. _ LIST

Daftar semua indeks.

Sintaksis

```
FT._LIST
```

Nilai yang ditampilkan

Mengembalikan array nama indeks

KAKI. ALIASADD

Tambahkan alias untuk indeks. Nama alias baru dapat digunakan di mana saja yang diperlukan nama indeks.

Sintaksis

```
FT.ALIASADD <alias> <index-name>
```

Nilai yang ditampilkan

Mengembalikan string pesan OK sederhana atau balasan kesalahan.

KAKI. ALIASDEL

Hapus alias yang ada untuk indeks.

Sintaksis

```
FT.ALIASDEL <alias>
```

Nilai yang ditampilkan

Mengembalikan string pesan OK sederhana atau balasan kesalahan.

KAKI. ALIASUPDATE

Perbarui alias yang ada untuk menunjuk ke indeks fisik yang berbeda. Perintah ini hanya mempengaruhi referensi future ke alias. Operasi yang sedang berlangsung (FT. SEARCH, FT. AGGREGATE) tidak terpengaruh oleh perintah ini.

Sintaksis

```
FT.ALIASUPDATE <alias> <index>
```

Nilai yang ditampilkan

Mengembalikan string pesan OK sederhana atau balasan kesalahan.

KAKI. _ ALIASLIST

Daftar alias indeks.

Sintaksis

```
FT._ALIASLIST
```

Nilai yang ditampilkan

Mengembalikan array ukuran jumlah alias saat ini. Setiap elemen dari array adalah pasangan alias-index.

KAKI. PROFILE

Jalankan kueri dan kembalikan informasi profil tentang kueri itu.

Sintaksis

```
FT.PROFILE  
  
<index>  
SEARCH | AGGREGATE  
[LIMITED]  
QUERY <query ....>
```

Nilai yang ditampilkan

Sebuah array dua elemen. Elemen pertama adalah hasil dari FT.AGGREGATE perintah FT.SEARCH atau yang diprofilkan. Elemen kedua adalah array informasi kinerja dan profil.

KAKI. EXPLAIN

Mengurai kueri dan mengembalikan informasi tentang bagaimana kueri itu diurai.

Sintaksis

```
FT.EXPLAIN <index> <query>
```

Nilai yang ditampilkan

String yang berisi hasil yang diuraikan.

KAKI. EXPLAINCLI

Sama seperti FT.EXPLAIN perintah kecuali bahwa hasilnya ditampilkan dalam format yang berbeda lebih berguna dengan redis-cli.

Sintaksis

```
FT.EXPLAINCLI <index> <query>
```

Nilai yang ditampilkan

String yang berisi hasil yang diuraikan.

Keamanan di MemoryDB

Keamanan cloud di AWS adalah prioritas tertinggi. Sebagai AWS pelanggan, Anda mendapat manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan organisasi yang paling sensitif terhadap keamanan.

Keamanan adalah tanggung jawab bersama antara Anda AWS dan Anda. [Model tanggung jawab bersama](#) menjelaskan hal ini sebagai keamanan cloud dan keamanan dalam cloud:

- Keamanan cloud — AWS bertanggung jawab untuk melindungi infrastruktur yang menjalankan AWS layanan di AWS Cloud. AWS juga memberi Anda layanan yang dapat Anda gunakan dengan aman. Auditor pihak ketiga secara teratur menguji dan memverifikasi efektivitas keamanan kami sebagai bagian dari [Program AWS Kepatuhan Program AWS Kepatuhan](#) . Untuk mempelajari tentang program kepatuhan yang berlaku untuk MemoryDB, lihat [AWS Layanan dalam Lingkup oleh AWS Layanan Program Kepatuhan dalam Lingkup oleh Program](#) .
- Keamanan di cloud — Tanggung jawab Anda ditentukan oleh AWS layanan yang Anda gunakan. Anda juga bertanggung jawab atas faktor lain, yang mencakup kepekaan data Anda, persyaratan perusahaan, serta peraturan perundangan yang berlaku

Dokumentasi ini membantu Anda memahami cara menerapkan model tanggung jawab bersama saat menggunakan MemoryDB. Ini menunjukkan kepada Anda cara mengkonfigurasi MemoryDB untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga belajar cara menggunakan AWS layanan lain yang membantu Anda memantau dan mengamankan sumber daya MemoryDB Anda.

Daftar Isi

- [Perlindungan data di MemoryDB](#)
- [Manajemen identitas dan akses di MemoryDB](#)
- [Pencatatan dan pemantauan](#)
- [Validasi kepatuhan untuk MemoryDB](#)
- [Keamanan infrastruktur di MemoryDB](#)
- [Privasi lalu lintas antar jaringan](#)
- [Pembaruan layanan di MemoryDB](#)

Perlindungan data di MemoryDB

[Model tanggung jawab AWS bersama model](#) berlaku untuk perlindungan data di. Seperti yang dijelaskan dalam model AWS ini, bertanggung jawab untuk melindungi infrastruktur global yang menjalankan semua AWS Cloud. Anda bertanggung jawab untuk mempertahankan kendali atas konten yang di-host pada infrastruktur ini. Anda juga bertanggung jawab atas tugas-tugas konfigurasi dan manajemen keamanan untuk AWS layanan yang Anda gunakan. Lihat informasi yang lebih lengkap tentang privasi data dalam [Pertanyaan Umum Privasi Data](#). Lihat informasi tentang perlindungan data di Eropa di pos blog [Model Tanggung Jawab Bersama dan GDPR AWS](#) di Blog Keamanan AWS .

Untuk tujuan perlindungan data, kami menyarankan Anda melindungi Akun AWS kredensial dan mengatur pengguna individu dengan AWS IAM Identity Center atau AWS Identity and Access Management (IAM). Dengan cara itu, setiap pengguna hanya diberi izin yang diperlukan untuk memenuhi tanggung jawab tugasnya. Kami juga menyarankan supaya Anda mengamankan data dengan cara-cara berikut:

- Gunakan autentikasi multi-faktor (MFA) pada setiap akun.
- Gunakan SSL/TLS untuk berkomunikasi dengan sumber daya. AWS Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Siapkan API dan pencatatan aktivitas pengguna dengan AWS CloudTrail.
- Gunakan solusi AWS enkripsi, bersama dengan semua kontrol keamanan default di dalamnya AWS layanan.
- Gunakan layanan keamanan terkelola lanjut seperti Amazon Macie, yang membantu menemukan dan mengamankan data sensitif yang disimpan di Amazon S3.
- Jika Anda memerlukan modul kriptografi tervalidasi FIPS 140-2 saat mengakses AWS melalui antarmuka baris perintah atau API, gunakan titik akhir FIPS. Lihat informasi yang lebih lengkap tentang titik akhir FIPS yang tersedia di [Standar Pemrosesan Informasi Federal \(FIPS\) 140-2](#).

Kami sangat merekomendasikan agar Anda tidak pernah memasukkan informasi identifikasi yang sensitif, seperti nomor rekening pelanggan Anda, ke dalam tanda atau bidang isian bebas seperti bidang Nama. Ini termasuk saat Anda bekerja dengan atau lainnya AWS layanan menggunakan konsol, API AWS CLI, atau AWS SDK. Data apa pun yang Anda masukkan ke dalam tanda atau bidang isian bebas yang digunakan untuk nama dapat digunakan untuk log penagihan atau log diagnostik. Saat Anda memberikan URL ke server eksternal, kami sangat menganjurkan supaya

Anda tidak menyertakan informasi kredensial di dalam URL untuk memvalidasi permintaan Anda ke server itu.

Keamanan data di MemoryDB

Untuk membantu menjaga keamanan data Anda, MemoryDB dan Amazon EC2 menyediakan mekanisme untuk melindungi terhadap akses data Anda yang tidak sah di server.

MemoryDB juga menyediakan fitur enkripsi untuk data pada cluster:

- Enkripsi in-transit mengenkripsi data Anda setiap kali data bergerak dari satu tempat ke tempat lain, misalnya antara beberapa simpul di klaster Anda atau antara klaster dan aplikasi Anda.
- Enkripsi AT-rest mengenkripsi log transaksi dan data on-disk Anda selama operasi snapshot.

Anda juga dapat menggunakan [Mengautentikasi pengguna dengan Daftar Kontrol Akses \(ACL\)](#) untuk mengontrol akses pengguna ke cluster Anda.

Topik

- [Enkripsi At-Rest di MemoryDB](#)
- [Enkripsi dalam transit \(TLS\) di MemoryDB](#)
- [Mengautentikasi pengguna dengan Daftar Kontrol Akses \(ACL\)](#)
- [Autentikasi dengan IAM](#)

Enkripsi At-Rest di MemoryDB

Untuk membantu menjaga keamanan data Anda, MemoryDB dan Amazon S3 menyediakan berbagai cara untuk membatasi akses ke data di cluster Anda. Untuk informasi selengkapnya, lihat [MemoryDB dan Amazon VPC](#) dan [Manajemen identitas dan akses di MemoryDB](#).

Enkripsi saat istirahat MemoryDB selalu diaktifkan untuk meningkatkan keamanan data dengan mengenkripsi data persisten. Ini mengenkripsi aspek-aspek berikut:

- Data dalam log transaksi
- Disk selama operasi sinkronisasi, snapshot, dan swap
- Snapshot disimpan di Amazon S3

MemoryDB menawarkan enkripsi default (dikelola layanan) saat istirahat, serta kemampuan untuk menggunakan kunci root pelanggan yang dikelola pelanggan simetris Anda sendiri di [AWS Key Management Service \(KMS\)](#).

Data yang disimpan di SSD (solid-state drive) dalam cluster yang diaktifkan tingkat data selalu dienkripsi secara default.

Untuk informasi tentang enkripsi in-transit, lihat [Enkripsi dalam transit \(TLS\) di MemoryDB](#)

Topik

- [Menggunakan Customer Managed Keys dari AWS KMS](#)
- [Lihat Juga](#)

Menggunakan Customer Managed Keys dari AWS KMS

MemoryDB mendukung kunci root terkelola pelanggan simetris (kunci KMS) untuk enkripsi saat istirahat. Kunci KMS yang dikelola pelanggan adalah kunci enkripsi yang Anda buat, miliki, dan kelola di akun Anda. AWS Untuk informasi selengkapnya, lihat [Kunci Root Pelanggan](#) di Panduan Pengembang Layanan Manajemen AWS Kunci. Kunci harus dibuat di AWS KMS sebelum dapat digunakan dengan MemoryDB.

Untuk mempelajari cara membuat kunci root AWS KMS, lihat [Membuat Kunci](#) di Panduan Pengembang Layanan Manajemen AWS Kunci.

MemoryDB memungkinkan Anda untuk berintegrasi dengan AWS KMS. Untuk informasi selengkapnya, lihat [Menggunakan Grant](#) dalam Panduan Developer AWS Key Management Service. Tidak diperlukan tindakan pelanggan untuk mengaktifkan integrasi MemoryDB dengan AWS KMS.

Kunci `kms:ViaService` kondisi membatasi penggunaan kunci AWS KMS untuk permintaan dari AWS layanan tertentu. Untuk digunakan `kms:ViaService` dengan MemoryDB, sertakan kedua `ViaService` nama dalam nilai kunci kondisi: `memorydb.amazonaws.com` Untuk informasi lebih lanjut, lihat [kms: ViaService](#).

Anda dapat menggunakan [AWS CloudTrail](#) untuk melacak permintaan yang dikirimkan MemoryDB AWS Key Management Service atas nama Anda. Semua panggilan API yang AWS Key Management Service terkait dengan kunci yang dikelola pelanggan memiliki CloudTrail log yang sesuai. Anda juga dapat melihat hibah yang dibuat oleh MemoryDB dengan memanggil panggilan API [ListGrants](#) KMS.

Setelah cluster dienkripsi menggunakan kunci yang dikelola pelanggan, semua snapshot untuk cluster dienkripsi sebagai berikut:

- Snapshot harian otomatis dienkripsi menggunakan kunci yang dikelola pelanggan yang terkait dengan cluster.
- Snapshot akhir yang dibuat saat cluster dihapus, juga dienkripsi menggunakan kunci terkelola pelanggan yang terkait dengan cluster.
- Snapshot yang dibuat secara manual dienkripsi secara default untuk menggunakan kunci KMS yang terkait dengan cluster. Anda dapat menggantinya dengan memilih kunci dikelola pelanggan yang lain.
- Menyalin snapshot secara default menggunakan kunci terkelola pelanggan yang terkait dengan snapshot sumber. Anda dapat menggantinya dengan memilih kunci dikelola pelanggan yang lain.

Note

- Kunci terkelola pelanggan tidak dapat digunakan saat mengekspor snapshot ke bucket Amazon S3 yang Anda pilih. [Namun, semua snapshot yang diekspor ke Amazon S3 dienkripsi menggunakan enkripsi sisi Server](#). Anda dapat memilih untuk menyalin file snapshot ke objek S3 baru dan mengenkripsi menggunakan kunci KMS yang dikelola pelanggan, menyalin file ke bucket S3 lain yang diatur dengan enkripsi default menggunakan kunci KMS atau mengubah opsi enkripsi dalam file itu sendiri.
- Anda juga dapat menggunakan kunci yang dikelola pelanggan untuk mengenkripsi snapshot yang dibuat secara manual yang tidak menggunakan kunci yang dikelola

pelanggan untuk enkripsi. Dengan opsi ini, file snapshot yang disimpan di Amazon S3 dienkripsi menggunakan kunci KMS, meskipun data tidak dienkripsi pada cluster asli.

Memulihkan dari snapshot memungkinkan Anda memilih dari opsi enkripsi yang tersedia, mirip dengan pilihan enkripsi yang tersedia saat membuat cluster baru.

- Jika Anda menghapus kunci atau [menonaktifkan](#) kunci dan [mencabut hibah](#) untuk kunci yang Anda gunakan untuk mengenkripsi klaster, cluster menjadi tidak dapat dipulihkan. Dengan kata lain, itu tidak dapat dimodifikasi atau dipulihkan setelah kegagalan perangkat keras. AWS KMS menghapus kunci root hanya setelah masa tunggu setidaknya tujuh hari. Setelah kunci dihapus, Anda dapat menggunakan kunci yang dikelola pelanggan yang berbeda untuk membuat snapshot untuk tujuan arsip.
- Rotasi kunci otomatis mempertahankan properti kunci root AWS KMS Anda, sehingga rotasi tidak berpengaruh pada kemampuan Anda untuk mengakses data MemoryDB Anda. Cluster MemoryDB terenkripsi tidak mendukung rotasi kunci manual, yang melibatkan pembuatan kunci root baru dan memperbarui referensi apa pun ke kunci lama. Untuk mempelajari selengkapnya, lihat [Memutar Kunci Root Pelanggan](#) di Panduan Pengembang Layanan Manajemen AWS Kunci.
- Mengenkripsi cluster MemoryDB menggunakan kunci KMS memerlukan satu hibah per cluster. Hibah ini digunakan sepanjang umur cluster. Selain itu, satu hibah per snapshot digunakan selama pembuatan snapshot. Hibah ini dihentikan setelah snapshot dibuat.
- Untuk informasi selengkapnya tentang AWS hibah dan batasan KMS, lihat [Kuota di Panduan Pengembang Layanan Manajemen AWS Utama](#).

Lihat Juga

- [Enkripsi dalam transit \(TLS\) di MemoryDB](#)
- [MemoryDB dan Amazon VPC](#)
- [Manajemen identitas dan akses di MemoryDB](#)

Enkripsi dalam transit (TLS) di MemoryDB

Untuk membantu menjaga keamanan data Anda, MemoryDB dan Amazon EC2 menyediakan mekanisme untuk melindungi terhadap akses data Anda yang tidak sah di server. Dengan menyediakan kemampuan enkripsi dalam perjalanan, MemoryDB memberi Anda alat yang dapat

Anda gunakan untuk membantu melindungi data Anda saat berpindah dari satu lokasi ke lokasi lain. Misalnya, Anda dapat memindahkan data dari node utama ke node replika baca di dalam klaster, atau antara cluster dan aplikasi Anda.

Topik

- [Gambaran umum enkripsi bergerak](#)
- [Lihat juga](#)

Gambaran umum enkripsi bergerak

Enkripsi dalam transit MemoryDB adalah fitur yang meningkatkan keamanan data Anda pada titik yang paling rentan — saat transit dari satu lokasi ke lokasi lain.

Enkripsi dalam transit MemoryDB mengimplementasikan fitur-fitur berikut:

- Koneksi terenkripsi — baik koneksi server dan klien dienkripsi Transport Layer Security (TLS).
- Replikasi terenkripsi— data yang bergerak antara node primer dan node replika dienkripsi.
- Autentikasi server—Klien dapat mengautentikasi bahwa koneksinya dilakukan ke server yang benar.

Mulai 07/20/2023, TLS 1.2 adalah versi minimum yang didukung untuk cluster baru dan yang sudah ada. Gunakan [tautan](#) ini untuk mempelajari lebih lanjut tentang TLS 1.2 di AWS.

Untuk informasi lebih lanjut tentang menghubungkan ke cluster MemoryDB, lihat. [Menghubungkan ke node MemoryDB menggunakan redis-cli](#)

Lihat juga

- [Enkripsi At-Rest di MemoryDB](#)
- [Mengautentikasi Pengguna dengan Daftar Kontrol Akses \(ACL\)](#)
- [MemoryDB dan Amazon VPC](#)
- [Manajemen identitas dan akses di MemoryDB](#)

Mengautentikasi pengguna dengan Daftar Kontrol Akses (ACL)

Anda dapat mengautentikasi pengguna dengan Access Control List (ACL).

ACL memungkinkan Anda mengontrol akses kluster dengan mengelompokkan pengguna. Daftar kontrol Access ini dirancang sebagai cara untuk mengatur akses ke cluster.

Dengan ACL, Anda membuat pengguna dan menetapkan mereka izin tertentu dengan menggunakan string akses, seperti yang dijelaskan di bagian berikutnya. Anda menetapkan pengguna ke daftar kontrol Access selaras dengan peran tertentu (administrator, sumber daya manusia) yang kemudian disebar ke satu atau beberapa cluster MemoryDB. Dengan melakukan ini, Anda dapat menetapkan batas keamanan antara klien menggunakan cluster atau cluster MemoryDB yang sama dan mencegah klien mengakses data satu sama lain.

ACL dirancang untuk mendukung pengenalan [Redis ACL di Redis OSS 6](#). Saat Anda menggunakan ACL dengan cluster MemoryDB Anda, ada beberapa batasan:

- Anda tidak dapat menentukan kata sandi dalam string akses. Anda mengatur kata sandi dengan [CreateUser](#) atau [UpdateUser](#) panggilan.
- Untuk hak pengguna, Anda meneruskan on dan off sebagai bagian dari string akses. Jika keduanya tidak ditentukan dalam string akses, pengguna ditugaskan off dan tidak memiliki hak akses ke cluster.
- Anda tidak dapat menggunakan perintah terlarang. Jika Anda menentukan perintah terlarang, pengecualian akan dilemparkan. Untuk daftar perintah tersebut, lihat [Perintah Redis OSS Terbatas](#).
- Anda tidak dapat menggunakan perintah `reset` sebagai bagian dari string akses. Anda menentukan kata sandi dengan parameter API, dan MemoryDB mengelola kata sandi. Dengan demikian, Anda tidak dapat menggunakan `reset` karena akan menghapus semua kata sandi untuk pengguna.
- Redis OSS 6 memperkenalkan perintah [ACL LIST](#). Perintah ini menampilkan daftar pengguna beserta aturan ACL yang berlaku untuk setiap pengguna. MemoryDB mendukung `ACL LIST` perintah, tetapi tidak menyertakan dukungan untuk hash kata sandi seperti yang dilakukan Redis OSS. Dengan MemoryDB, Anda dapat menggunakan [DescribeUsers](#) operasi untuk mendapatkan informasi serupa, termasuk aturan yang terkandung dalam string akses. Namun, [DescribeUser](#) tidak mengambil kata sandi pengguna.

[Perintah read-only lainnya yang didukung oleh MemoryDB termasuk ACL WHOAMI, ACL USERS, dan ACL CAT](#). MemoryDB tidak mendukung perintah ACL berbasis tulis lainnya.

Menggunakan ACL dengan MemoryDB dijelaskan secara lebih rinci berikut.

Topik

- [Menentukan Izin Menggunakan String Akses](#)
- [Kemampuan pencarian vektor](#)
- [Menerapkan ACL ke cluster untuk MemoryDB](#)

Menentukan Izin Menggunakan String Akses

Untuk menentukan izin ke cluster MemoryDB, Anda membuat string akses dan menentukannya ke pengguna, menggunakan salah satu atau. AWS CLI AWS Management Console

String akses didefinisikan sebagai daftar aturan yang dipisahkan spasi yang diterapkan pada pengguna. String akses menentukan perintah yang dapat dijalankan oleh pengguna dan kunci yang dapat dioperasikan oleh pengguna. Untuk dapat menjalankan perintah, pengguna harus memiliki akses ke perintah yang dijalankan dan semua kunci yang diakses oleh perintah itu. Aturan diterapkan dari kiri ke kanan secara kumulatif, dan string yang lebih sederhana dapat digunakan sebagai pengganti yang disediakan jika ada kelebihan dalam string yang disediakan.

Untuk informasi tentang sintaks aturan ACL, lihat [ACL](#).

Pada contoh berikut, string akses merepresentasikan pengguna aktif dengan akses ke semua kunci dan perintah yang tersedia.

```
on ~* &* +@all
```

Sintaksis string akses diuraikan sebagai berikut:

- on – Pengguna adalah pengguna yang aktif.
- ~* – Akses diberikan ke semua kunci yang tersedia.
- &*— Akses diberikan ke semua saluran pubsub.
- +@all – Akses diberikan ke semua perintah yang tersedia.

Pengaturan di atas adalah pengaturan yang tidak terlalu membatasi. Anda dapat mengubah pengaturan ini untuk membuatnya lebih aman.

Pada contoh berikut, string akses merepresentasikan pengguna dengan akses yang dibatasi untuk akses baca pada kunci yang diawali dengan ruang kunci "app::"

```
on ~app::* -@all +@read
```

Anda dapat mempersempit izin ini lebih lanjut dengan menampilkan daftar perintah yang dapat diakses pengguna:

+*command1* – Akses pengguna ke perintah dibatasi pada *command1*.

+*@category* – Akses pengguna dibatasi pada kategori perintah.

Untuk informasi tentang cara menetapkan string akses ke pengguna, lihat [Membuat Pengguna dan Daftar Kontrol Akses dengan Konsol dan CLI](#).

Jika Anda memigrasikan beban kerja yang ada ke MemoryDB, Anda dapat mengambil string akses dengan menelepon `ACL LIST`, tidak termasuk pengguna dan hash kata sandi apa pun.

Kemampuan pencarian vektor

Note

Fitur ini dalam rilis pratinjau untuk MemoryDB dan dapat berubah.

Untuk [Pencarian vektor](#), semua perintah pencarian termasuk dalam `@search` kategori dan kategori yang ada `@read@write`, `@fast` dan `@slow` diperbarui untuk menyertakan perintah pencarian. Jika pengguna tidak memiliki akses ke kategori, maka pengguna tidak memiliki akses ke perintah apa pun dalam kategori tersebut. Misalnya, jika pengguna tidak memiliki akses ke `@search`, maka pengguna tidak dapat menjalankan perintah terkait pencarian.

Tabel berikut menunjukkan pemetaan perintah pencarian ke kategori yang sesuai.

Perintah VSS	@read	@write	@fast	@slow
FT.CREATE		T	Y	
FT.DROPINDEX		Y	Y	
FT.LIST	Y			Y
FT.INFO	Y		Y	

Perintah VSS	@read	@write	@fast	@slow
FT.SEARCH	Y			Y
FT.AGGREGATE	Y			Y
FT.PROFILE	Y			Y
FT.ALIASADD		Y	Y	
FT.ALIASDEL		Y	Y	
FT.ALIASUPDATE		Y	Y	
FT._ALIASLIST	Y			Y
FT.EXPLAIN	Y		Y	
FT.EXPLAINCLI	Y		Y	
FT.CONFIG	Y		T	

Menerapkan ACL ke cluster untuk MemoryDB

Untuk menggunakan MemoryDB ACL, Anda mengambil langkah-langkah berikut:

1. Buat satu atau beberapa pengguna.
2. Buat ACL dan tambahkan pengguna ke daftar.

3. Tetapkan ACL ke cluster.

Langkah ini dijelaskan secara mendetail di bagian berikut.

Topik

- [Membuat Pengguna dan Daftar Kontrol Akses dengan Konsol dan CLI](#)
- [Mengelola Daftar Kontrol Akses dengan Konsol dan CLI](#)
- [Menetapkan daftar kontrol Akses ke cluster](#)

Membuat Pengguna dan Daftar Kontrol Akses dengan Konsol dan CLI

Informasi pengguna untuk pengguna ACL adalah nama pengguna, dan secara opsional kata sandi dan string akses. String akses menyediakan tingkat izin pada kunci dan perintah. Nama ini unik untuk pengguna dan itulah yang diteruskan ke mesin.

Pastikan bahwa izin pengguna yang Anda berikan masuk akal dengan tujuan ACL yang dimaksudkan. Misalnya, jika Anda membuat ACL yang dipanggil `Administrators`, setiap pengguna yang Anda tambahkan ke grup itu harus memiliki string aksesnya disetel ke akses penuh ke kunci dan perintah. Untuk pengguna di `e-commerce` ACL, Anda dapat mengatur string akses mereka ke akses hanya-baca.

MemoryDB secara otomatis mengkonfigurasi pengguna default per akun dengan nama pengguna. `"default"` Ini tidak akan dikaitkan dengan cluster apa pun kecuali eksplisit ditambahkan ke ACL. Anda tidak dapat mengubah atau menghapus pengguna ini. Pengguna ini dimaksudkan untuk kompatibilitas dengan perilaku default versi Redis OSS sebelumnya dan memiliki string akses yang memungkinkannya untuk memanggil semua perintah dan mengakses semua kunci.

ACL “akses terbuka” yang tidak dapat diubah akan dibuat untuk setiap akun yang berisi pengguna default. Ini adalah satu-satunya ACL pengguna default dapat menjadi anggota. Saat Anda membuat cluster, Anda harus memilih ACL untuk dikaitkan dengan cluster. Meskipun Anda memiliki opsi untuk menerapkan ACL “akses terbuka” dengan pengguna default, kami sangat menyarankan untuk membuat ACL dengan pengguna yang memiliki izin terbatas pada kebutuhan bisnis mereka.

Cluster yang tidak mengaktifkan TLS harus menggunakan ACL “akses terbuka” untuk memberikan otentikasi terbuka.

ACL dapat dibuat tanpa pengguna. ACL kosong tidak akan memiliki akses ke cluster dan hanya dapat dikaitkan dengan cluster yang mendukung TLS.

Saat membuat pengguna, Anda dapat menetapkan hingga dua kata sandi. Saat Anda memodifikasi kata sandi, koneksi apa pun yang ada ke cluster dipertahankan.

Secara khusus, perhatikan kendala kata sandi pengguna ini saat menggunakan ACL untuk MemoryDB:

- Kata sandi harus terdiri dari 16–128 karakter yang dapat dicetak.
- Karakter non-alfanumerik berikut tidak diizinkan: , " " / @.

Mengelola Pengguna dengan Konsol dan CLI

Membuat pengguna (Konsol)

Untuk membuat pengguna di konsol

1. [Masuk ke AWS Management Console dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/.](https://console.aws.amazon.com/memorydb/)
2. Di panel navigasi kiri, pilih Pengguna.
3. Pilih Buat pengguna
4. Pada halaman Buat pengguna, masukkan Nama.

Batasan penamaan klaster adalah sebagai berikut:

- Harus berisi 1–40 karakter alfanumerik atau tanda hubung.
 - Harus diawali dengan huruf.
 - Tidak boleh berisi dua tanda hubung berurutan.
 - Tidak boleh diakhiri dengan tanda hubung.
5. Di bawah Kata Sandi, Anda dapat memasukkan hingga dua kata sandi.
 6. Di bawah Access string, masukkan string akses. String akses menetapkan tingkat izin untuk kunci dan perintah yang boleh digunakan pengguna.
 7. Untuk Tag, Anda dapat menerapkan tag secara opsional untuk mencari dan memfilter pengguna Anda atau melacak AWS biaya Anda.
 8. Pilih Buat.

Membuat pengguna menggunakan AWS CLI

Untuk membuat pengguna dengan menggunakan CLI

- Gunakan perintah [create-user](#) untuk membuat pengguna.

Untuk Linux, macOS, atau Unix:

```
aws memorydb create-user \  
  --user-name user-name-1 \  
  --access-string "~objects:* ~items:* ~public:*" \  
  --authentication-mode \  
    Passwords="abc",Type=password
```

Untuk Windows:

```
aws memorydb create-user ^  
  --user-name user-name-1 ^  
  --access-string "~objects:* ~items:* ~public:*" ^  
  --authentication-mode \  
    Passwords="abc",Type=password
```

Memodifikasi pengguna (Konsol)

Untuk memodifikasi pengguna di konsol

1. [Masuk ke AWS Management Console dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/.](https://console.aws.amazon.com/memorydb/)
2. Di panel navigasi kiri, pilih Pengguna.
3. Pilih tombol radio di sebelah pengguna yang ingin Anda modifikasi dan kemudian pilih Tindakan -> Ubah
4. Jika Anda ingin mengubah kata sandi, pilih tombol radio Ubah kata sandi. Perhatikan bahwa jika Anda memiliki dua kata sandi, Anda harus memasukkan keduanya saat memodifikasi salah satunya.
5. Jika Anda memperbarui string akses, masukkan yang baru.
6. Pilih Ubah.

Memodifikasi pengguna menggunakan AWS CLI

Untuk mengubah pengguna menggunakan CLI

1. Gunakan perintah [update-user](#) untuk memodifikasi pengguna.
2. Ketika pengguna dimodifikasi, daftar kontrol Access yang terkait dengan pengguna diperbarui, bersama dengan cluster apa pun yang terkait dengan ACL. Semua koneksi yang ada akan dipertahankan. Berikut ini adalah beberapa contohnya.

Untuk Linux, macOS, atau Unix:

```
aws memorydb update-user \  
  --user-name user-name-1 \  
  --access-string "~objects:* ~items:* ~public:~"
```

Untuk Windows:

```
aws memorydb update-user ^  
  --user-name user-name-1 ^  
  --access-string "~objects:* ~items:* ~public:~"
```

Melihat detail pengguna (Konsol)

Untuk melihat detail pengguna di konsol

1. [Masuk ke AWS Management Console dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/](https://console.aws.amazon.com/memorydb/).
2. Di panel navigasi kiri, pilih Pengguna.
3. Pilih pengguna di bawah Nama pengguna atau gunakan kotak pencarian untuk menemukan pengguna.
4. Di bawah Pengaturan pengguna, Anda dapat meninjau string akses pengguna, jumlah kata sandi, status, dan Nama Sumber Daya Amazon (ARN).
5. Di bawah Daftar kontrol akses (ACL) Anda dapat meninjau ACL milik pengguna.
6. Di bawah Tag, Anda dapat meninjau tag apa pun yang terkait dengan pengguna.

Melihat detail pengguna menggunakan AWS CLI

Gunakan [perintah deskripsikan pengguna](#) untuk melihat detail pengguna.

```
aws memorydb describe-users \  
  --user-name my-user-name
```

Menghapus pengguna (Konsol)

Untuk menghapus pengguna di konsol

1. [Masuk ke AWS Management Console dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/.](https://console.aws.amazon.com/memorydb/)
2. Di panel navigasi kiri, pilih Pengguna.
3. Pilih tombol radio di sebelah pengguna yang ingin Anda modifikasi dan kemudian pilih Tindakan -> Hapus
4. Untuk mengonfirmasi, masukkan delete di kotak teks konfirmasi dan kemudian pilih Hapus.
5. Untuk membatalkan, pilih Batalkan.

Menghapus pengguna menggunakan AWS CLI

Untuk menghapus pengguna menggunakan CLI

- Gunakan perintah [delete-user](#) untuk menghapus pengguna.

Akun dihapus dan dihapus dari daftar kontrol Akses apa pun yang menjadi miliknya. Berikut adalah contohnya.

Untuk Linux, macOS, atau Unix:

```
aws memorydb delete-user \  
  --user-name user-name-2
```

Untuk Windows:

```
aws memorydb delete-user ^  
  --user-name user-name-2
```

Mengelola Daftar Kontrol Akses dengan Konsol dan CLI

Anda dapat membuat daftar kontrol Access untuk mengatur dan mengontrol akses pengguna ke satu atau beberapa cluster, seperti yang ditunjukkan berikut.

Gunakan prosedur berikut untuk mengelola daftar kontrol Access menggunakan konsol.

Membuat Daftar Kontrol Akses (ACL) (Konsol)

Untuk membuat daftar kontrol Access menggunakan konsol

1. [Masuk ke AWS Management Console dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/.](https://console.aws.amazon.com/memorydb/)
2. Di panel navigasi kiri, pilih Daftar kontrol akses (ACL).
3. Pilih Buat ACL.
4. Pada halaman Create Access Control List (ACL), masukkan nama ACL.

Batasan penamaan klaster adalah sebagai berikut:

- Harus berisi 1–40 karakter alfanumerik atau tanda hubung.
 - Harus diawali dengan huruf.
 - Tidak boleh berisi dua tanda hubung berurutan.
 - Tidak boleh diakhiri dengan tanda hubung.
5. Di bawah Pengguna terpilih melakukan salah satu hal berikut:
 - a. Buat pengguna baru dengan memilih Buat pengguna
 - b. Tambahkan pengguna dengan memilih Kelola lalu pilih pengguna dari dialog Kelola pengguna lalu pilih Pilih.
 6. Untuk Tag, Anda dapat menerapkan tag secara opsional untuk mencari dan memfilter ACL Anda atau melacak biaya Anda AWS .
 7. Pilih Buat.

Membuat Access Control List (ACL) menggunakan AWS CLI

Gunakan prosedur berikut untuk membuat daftar kontrol Akses menggunakan CLI.

Untuk membuat ACL baru dan menambahkan pengguna dengan menggunakan CLI

- Gunakan perintah [create-acl untuk membuat ACL](#).

Untuk Linux, macOS, atau Unix:

```
aws memorydb create-acl \  
  --acl-name "new-acl-1" \  
  --user-names "user-name-1" "user-name-2"
```

Untuk Windows:

```
aws memorydb create-acl ^  
  --acl-name "new-acl-1" ^  
  --user-names "user-name-1" "user-name-2"
```

Memodifikasi Daftar Kontrol Akses (ACL) (konsol)

Untuk mengubah daftar kontrol Access menggunakan konsol

1. [Masuk ke AWS Management Console dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/](https://console.aws.amazon.com/memorydb/).
2. Di panel navigasi kiri, pilih Daftar kontrol akses (ACL).
3. Pilih ACL yang ingin Anda ubah dan kemudian pilih Modify
4. Pada halaman Ubah, di bawah Pengguna yang dipilih lakukan salah satu hal berikut:
 - a. Buat pengguna baru dengan memilih Buat pengguna untuk ditambahkan ke ACL.
 - b. Menambah atau menghapus pengguna dengan memilih Kelola lalu memilih atau menghapus pilihan pengguna dari dialog Kelola pengguna lalu memilih Pilih.
5. Pada halaman Create Access Control List (ACL), masukkan nama ACL.

Batasan penamaan klaster adalah sebagai berikut:

- Harus berisi 1–40 karakter alfanumerik atau tanda hubung.
- Harus diawali dengan huruf.
- Tidak boleh berisi dua tanda hubung berurutan.
- Tidak boleh diakhiri dengan tanda hubung.

6. Di bawah Pengguna terpilih melakukan salah satu hal berikut:
 - a. Buat pengguna baru dengan memilih Buat pengguna
 - b. Tambahkan pengguna dengan memilih Kelola lalu pilih pengguna dari dialog Kelola pengguna lalu pilih Pilih.
7. Pilih Ubah untuk menyimpan perubahan Anda atau Batalkan untuk membuangnya.

Memodifikasi Access Control List (ACL) menggunakan AWS CLI

Untuk memodifikasi ACL dengan menambahkan pengguna baru atau menghapus anggota saat ini dengan menggunakan CLI

- Gunakan perintah [update-acl untuk memodifikasi ACL](#).

Untuk Linux, macOS, atau Unix:

```
aws memorydb update-acl --acl-name new-acl-1 \  
--user-names-to-add user-name-3 \  
--user-names-to-remove user-name-2
```

Untuk Windows:

```
aws memorydb update-acl --acl-name new-acl-1 ^  
--user-names-to-add user-name-3 ^  
--user-names-to-remove user-name-2
```

Note

Setiap koneksi terbuka milik pengguna yang dihapus dari ACL diakhiri dengan perintah ini.

Melihat detail Daftar Kontrol Akses (ACL) (Konsol)

Untuk melihat detail ACL di konsol

1. [Masuk ke AWS Management Console dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/.](https://console.aws.amazon.com/memorydb/)
2. Di panel navigasi kiri, pilih Daftar kontrol akses (ACL).

3. Pilih ACL di bawah nama ACL atau gunakan kotak pencarian untuk menemukan ACL.
4. Di bawah Pengguna, Anda dapat meninjau daftar pengguna yang terkait dengan ACL.
5. Di bawah Kluster terkait, Anda dapat meninjau kluster tempat ACL berada.
6. Di bawah Tag Anda dapat meninjau tag apa pun yang terkait dengan ACL.

Melihat Daftar Kontrol Akses (ACL) menggunakan AWS CLI

Gunakan perintah [describe-acls](#) untuk melihat detail ACL.

```
aws memorydb describe-acls \  
--acl-name test-group
```

Menghapus Daftar Kontrol Akses (ACL) (konsol)

Untuk menghapus daftar kontrol Access menggunakan konsol

1. [Masuk ke AWS Management Console dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/.](https://console.aws.amazon.com/memorydb/)
2. Di panel navigasi kiri, pilih Daftar kontrol akses (ACL).
3. Pilih ACL yang ingin Anda ubah dan kemudian pilih Hapus
4. Pada halaman Hapus, masukkan delete di kotak konfirmasi dan pilih Hapus atau Batal untuk menghindari penghapusan ACL.

ACL itu sendiri, bukan pengguna yang termasuk dalam grup, dihapus.

Menghapus Access Control List (ACL) menggunakan AWS CLI

Untuk menghapus ACL dengan menggunakan CLI

- Gunakan perintah [delete-acl](#) untuk menghapus ACL.

Untuk Linux, macOS, atau Unix:

```
aws memorydb delete-acl \  
--acl-name
```

Untuk Windows:

```
aws memorydb delete-acl ^  
  --acl-name
```

Contoh sebelumnya menghasilkan respons berikut.

```
aws memorydb delete-acl --acl-name "new-acl-1"  
{  
  "ACLName": "new-acl-1",  
  "Status": "deleting",  
  "EngineVersion": "6.2",  
  "UserNames": [  
    "user-name-1",  
    "user-name-3"  
  ],  
  "clusters": [],  
  "ARN": "arn:aws:memorydb:us-east-1:493071037918:acl/new-acl-1"  
}
```

Menetapkan daftar kontrol Akses ke cluster

Setelah Anda membuat ACL dan menambahkan pengguna, langkah terakhir dalam menerapkan ACL adalah menetapkan ACL ke cluster.

Menetapkan daftar kontrol Access ke cluster Menggunakan Konsol

Untuk menambahkan ACL ke cluster menggunakan AWS Management Console, lihat [Membuat cluster MemoryDB](#).

Menetapkan daftar kontrol Access ke cluster Menggunakan AWS CLI

AWS CLI Operasi berikut membuat cluster dengan enkripsi dalam transit (TLS) diaktifkan dan `acl-name` parameter dengan nilai `my-acl-name`. Ganti grup subnet `subnet-group` dengan grup subnet yang ada.

Parameter Kunci

- **--engine-version** Harus 6.2.
- **--tls-enabled**— Digunakan untuk otentikasi dan untuk mengaitkan ACL.
- **--acl-name**— Nilai ini menyediakan daftar kontrol Access yang terdiri dari pengguna dengan izin akses tertentu untuk cluster.

Untuk Linux, macOS, atau Unix:

```
aws memorydb create-cluster \  
  --cluster-name "new-cluster" \  
  --description "new-cluster" \  
  --engine-version "6.2" \  
  --node-type db.r6g.large \  
  --tls-enabled \  
  --acl-name "new-acl-1" \  
  --subnet-group-name "subnet-group"
```

Untuk Windows:

```
aws memorydb create-cluster ^  
  --cluster-name "new-cluster" ^  
  --cluster-description "new-cluster" ^  
  --engine-version "6.2" ^  
  --node-type db.r6g.large ^  
  --tls-enabled ^  
  --acl-name "new-acl-1" ^  
  --subnet-group-name "subnet-group"
```

AWS CLI Operasi berikut memodifikasi cluster dengan enkripsi dalam transit (TLS) diaktifkan dan acl-name parameter dengan nilai. new-acl-2

Untuk Linux, macOS, atau Unix:

```
aws memorydb update-cluster \  
  --cluster-name cluster-1 \  
  --acl-name "new-acl-2"
```

Untuk Windows:

```
aws memorydb update-cluster ^  
  --cluster-name cluster-1 ^  
  --acl-name "new-acl-2"
```

Autentikasi dengan IAM

Topik

- [Gambaran Umum](#)
- [Batasan](#)
- [Pengaturan](#)
- [Terhubung](#)

Gambaran Umum

Dengan Autentikasi IAM Anda dapat mengautentikasi koneksi ke MemoryDB menggunakan identitas AWS IAM, ketika cluster Anda dikonfigurasi untuk menggunakan Redis OSS versi 7 atau lebih tinggi. Hal ini memungkinkan Anda memperkuat model keamanan Anda dan menyederhanakan banyak tugas keamanan administratif. Dengan Autentikasi IAM Anda dapat mengonfigurasi kontrol akses berbutir halus untuk setiap cluster MemoryDB individu dan pengguna MemoryDB dan mengikuti prinsip izin hak istimewa paling sedikit. Otentikasi IAM untuk MemoryDB bekerja dengan menyediakan token otentikasi IAM berumur pendek alih-alih kata sandi pengguna MemoryDB yang berumur panjang di OSS Redis atau perintah. AUTH HELLO Untuk informasi selengkapnya tentang token autentikasi IAM, lihat [proses penandatanganan Signature Version 4](#) di Panduan Referensi AWS Umum dan contoh kode di bawah ini.

Anda dapat menggunakan identitas IAM dan kebijakan terkait mereka untuk lebih membatasi akses Redis OSS. Anda juga dapat memberikan akses ke pengguna dari penyedia Identitas federasi mereka langsung ke cluster MemoryDB.

Untuk menggunakan AWS IAM dengan MemoryDB, Anda harus terlebih dahulu membuat pengguna MemoryDB dengan mode otentikasi diatur ke IAM, kemudian Anda dapat membuat atau menggunakan kembali identitas IAM. Identitas IAM memerlukan kebijakan terkait untuk memberikan `memorydb:Connect` tindakan kepada cluster MemoryDB dan pengguna MemoryDB. Setelah dikonfigurasi, Anda dapat membuat token otentikasi IAM menggunakan AWS kredensial pengguna atau peran IAM. Akhirnya Anda perlu memberikan token otentikasi IAM berumur pendek sebagai kata sandi di klien Redis OSS Anda saat menghubungkan ke node cluster MemoryDB Anda. Klien Redis OSS dengan dukungan untuk penyedia kredensi dapat secara otomatis menghasilkan kredensi sementara secara otomatis untuk setiap koneksi baru. MemoryDB akan melakukan otentikasi IAM untuk permintaan koneksi pengguna MemoryDB yang mendukung IAM dan akan memvalidasi permintaan koneksi dengan IAM.

Batasan

Saat menggunakan autentikasi IAM, batasan berikut berlaku:

- Otentikasi IAM tersedia saat menggunakan mesin Redis OSS versi 7.0 atau lebih tinggi.
- Token autentikasi IAM berlaku selama 15 menit. Untuk koneksi yang berumur panjang, sebaiknya gunakan klien Redis OSS yang mendukung antarmuka penyedia kredensial.
- Koneksi yang diautentikasi IAM ke MemoryDB akan secara otomatis terputus setelah 12 jam. Koneksi dapat diperpanjang selama 12 jam dengan mengirim perintah AUTH atau HELLO dengan token autentikasi IAM baru.
- Autentikasi IAM tidak didukung dalam perintah MULTI EXEC.
- Saat ini, autentikasi IAM tidak mendukung semua kunci konteks kondisi global. Untuk informasi selengkapnya tentang kunci konteks kondisi global, lihat [Kunci konteks kondisi global AWS](#) dalam Panduan Pengguna IAM.

Pengaturan

Untuk mengatur autentikasi IAM:

1. Membuat klaster

```
aws memorydb create-cluster \  
  --cluster-name cluster-01 \  
  --description "MemoryDB IAM auth application" \  
  --node-type db.r6g.large \  
  --engine-version 7.0 \  
  --acl-name open-access
```

2. Buat dokumen kebijakan kepercayaan IAM, seperti yang ditunjukkan di bawah ini, untuk peran Anda yang memungkinkan akun Anda mengambil peran baru. Simpan kebijakan ini ke file bernama trust-policy.json.

```
{  
  "Version": "2012-10-17",  
  "Statement": {  
    "Effect": "Allow",  
    "Principal": { "AWS": "arn:aws:iam::123456789012:root" },  
    "Action": "sts:AssumeRole"  
  }  
}
```

3. Buat dokumen kebijakan IAM, seperti yang ditunjukkan di bawah ini. Simpan kebijakan ke file bernama policy.json.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect" : "Allow",
      "Action" : [
        "memorydb:connect"
      ],
      "Resource" : [
        "arn:aws:memorydb:us-east-1:123456789012:cluster/cluster-01",
        "arn:aws:memorydb:us-east-1:123456789012:user/iam-user-01"
      ]
    }
  ]
}
```

4. Buat peran IAM.

```
aws iam create-role \
  --role-name "memorydb-iam-auth-app" \
  --assume-role-policy-document file://trust-policy.json
```

5. Buat kebijakan IAM.

```
aws iam create-policy \
  --policy-name "memorydb-allow-all" \
  --policy-document file://policy.json
```

6. Lampirkan kebijakan IAM ke peran tersebut.

```
aws iam attach-role-policy \
  --role-name "memorydb-iam-auth-app" \
  --policy-arn "arn:aws:iam::123456789012:policy/memorydb-allow-all"
```

7. Buat pengguna baru yang didukung IAM.

```
aws memorydb create-user \
  --user-name iam-user-01 \
  --authentication-mode Type=iam \
  --access-string "on ~* +@all"
```

8. Buat ACL dan lampirkan pengguna.

```
aws memorydb create-acl \  
  --acl-name iam-acl-01 \  
  --user-names iam-user-01  
  
aws memorydb update-cluster \  
  --cluster-name cluster-01 \  
  --acl-name iam-acl-01
```

Terhubung

Terhubung dengan token sebagai kata sandi

Pertama-tama, Anda harus membuat token autentikasi IAM berumur pendek menggunakan [permintaan AWS SigV4 yang telah ditandatangani sebelumnya](#). Setelah itu Anda memberikan token otentikasi IAM sebagai kata sandi saat menghubungkan ke cluster MemoryDB, seperti yang ditunjukkan pada contoh di bawah ini.

```
String userName = "insert user name"  
String clusterName = "insert cluster name"  
String region = "insert region"  
  
// Create a default AWS Credentials provider.  
// This will look for AWS credentials defined in environment variables or system  
// properties.  
AWSCredentialsProvider awsCredentialsProvider = new  
  DefaultAWSCredentialsProviderChain();  
  
// Create an IAM authentication token request and signed it using the AWS credentials.  
// The pre-signed request URL is used as an IAM authentication token for MemoryDB.  
IAMAuthTokenRequest iamAuthTokenRequest = new IAMAuthTokenRequest(userName,  
  clusterName, region);  
String iamAuthToken =  
  iamAuthTokenRequest.toSignedRequestUri(awsCredentialsProvider.getCredentials());  
  
// Construct Redis OSS URL with IAM Auth credentials provider  
RedisURI redisURI = RedisURI.builder()  
  .withHost(host)  
  .withPort(port)  
  .withSsl(ssl)  
  .withAuthentication(userName, iamAuthToken)  
  .build();
```

```
// Create a new Lettuce Redis OSS client
RedisClusterClient client = RedisClusterClient.create(redisURI);
client.connect();
```

Di bawah ini adalah definisi untuk IAMAuthTokenRequest.

```
public class IAMAuthTokenRequest {
    private static final HttpMethodName REQUEST_METHOD = HttpMethodName.GET;
    private static final String REQUEST_PROTOCOL = "http://";
    private static final String PARAM_ACTION = "Action";
    private static final String PARAM_USER = "User";
    private static final String ACTION_NAME = "connect";
    private static final String SERVICE_NAME = "memorydb";
    private static final long TOKEN_EXPIRY_SECONDS = 900;

    private final String userName;
    private final String clusterName;
    private final String region;

    public IAMAuthTokenRequest(String userName, String clusterName, String region) {
        this.userName = userName;
        this.clusterName = clusterName;
        this.region = region;
    }

    public String toSignedRequestUri(AWSCredentials credentials) throws
    URISyntaxException {
        Request<Void> request = getSignableRequest();
        sign(request, credentials);
        return new URIBuilder(request.getEndpoint())
            .addParameters(toNamedValuePair(request.getParameters()))
            .build()
            .toString()
            .replace(REQUEST_PROTOCOL, "");
    }

    private <T> Request<T> getSignableRequest() {
        Request<T> request = new DefaultRequest<>(SERVICE_NAME);
        request.setHttpMethod(REQUEST_METHOD);
        request.setEndpoint(getRequestUri());
        request.addParameters(PARAM_ACTION, Collections.singletonList(ACTION_NAME));
        request.addParameters(PARAM_USER, Collections.singletonList(userName));
    }
}
```

```
        return request;
    }

    private URI getRequestUri() {
        return URI.create(String.format("%s%s/", REQUEST_PROTOCOL, clusterName));
    }

    private <T> void sign(SignableRequest<T> request, AWSCredentials credentials) {
        AWS4Signer signer = new AWS4Signer();
        signer.setRegionName(region);
        signer.setServiceName(SERVICE_NAME);

        DateTime dateTime = DateTime.now();
        dateTime = dateTime.plus(Duration.standardSeconds(TOKEN_EXPIRY_SECONDS));

        signer.presignRequest(request, credentials, dateTime.toDate());
    }

    private static List<NameValuePair> toNamedValuePair(Map<String, List<String>> in) {
        return in.entrySet().stream()
            .map(e -> new BasicNameValuePair(e.getKey(), e.getValue().get(0)))
            .collect(Collectors.toList());
    }
}
```

Terhubung dengan penyedia kredensial

Kode di bawah ini menunjukkan cara mengautentikasi dengan MemoryDB menggunakan penyedia kredensi otentikasi IAM.

```
String userName = "insert user name"
String clusterName = "insert cluster name"
String region = "insert region"

// Create a default AWS Credentials provider.
// This will look for AWS credentials defined in environment variables or system
// properties.
AWSCredentialsProvider awsCredentialsProvider = new
    DefaultAWSCredentialsProviderChain();

// Create an IAM authentication token request. Once this request is signed it can be
// used as an
// IAM authentication token for MemoryDB.
```

```
IAMAuthTokenRequest iamAuthTokenRequest = new IAMAuthTokenRequest(userName,
    clusterName, region);

// Create a Redis OSS credentials provider using IAM credentials.
RedisCredentialsProvider redisCredentialsProvider = new
    RedisIAMAuthCredentialsProvider(
        userName, iamAuthTokenRequest, awsCredentialsProvider);

// Construct Redis OSS URL with IAM Auth credentials provider
RedisURI redisURI = RedisURI.builder()
    .withHost(host)
    .withPort(port)
    .withSsl(ssl)
    .withAuthentication(redisCredentialsProvider)
    .build();

// Create a new Lettuce Redis OSS cluster client
RedisClusterClient client = RedisClusterClient.create(redisURI);
client.connect();
```

Di bawah ini adalah contoh klien cluster Lettuce Redis OSS yang membungkus IAM AuthTokenRequest dalam penyedia kredensial untuk menghasilkan kredensial sementara secara otomatis bila diperlukan.

```
public class RedisIAMAuthCredentialsProvider implements RedisCredentialsProvider {
    private static final long TOKEN_EXPIRY_SECONDS = 900;

    private final AWSCredentialsProvider awsCredentialsProvider;
    private final String userName;
    private final IAMAuthTokenRequest iamAuthTokenRequest;
    private final Supplier<String> iamAuthTokenSupplier;

    public RedisIAMAuthCredentialsProvider(String userName,
        IAMAuthTokenRequest iamAuthTokenRequest,
        AWSCredentialsProvider awsCredentialsProvider) {
        this.userName = userName;
        this.awsCredentialsProvider = awsCredentialsProvider;
        this.iamAuthTokenRequest = iamAuthTokenRequest;
        this.iamAuthTokenSupplier =
            Suppliers.memoizeWithExpiration(this::getIamAuthToken, TOKEN_EXPIRY_SECONDS,
                TimeUnit.SECONDS);
    }
}
```

```
@Override
public Mono<RedisCredentials> resolveCredentials() {
    return Mono.just(RedisCredentials.just(userName, iamAuthTokenSupplier.get()));
}

private String getIamAuthToken() {
    return
iamAuthTokenRequest.toSignedRequestUri(awsCredentialsProvider.getCredentials());
}
```

Manajemen identitas dan akses di MemoryDB

AWS Identity and Access Management (IAM) adalah AWS layanan yang membantu administrator mengontrol akses ke AWS sumber daya dengan aman. Administrator IAM mengontrol siapa yang dapat diautentikasi (masuk) dan diberi wewenang (memiliki izin) untuk menggunakan sumber daya MemoryDB. IAM adalah AWS layanan yang dapat Anda gunakan tanpa biaya tambahan.

Topik

- [Audiens](#)
- [Mengautentikasi dengan identitas](#)
- [Mengelola akses menggunakan kebijakan](#)
- [Bagaimana MemoryDB bekerja dengan IAM](#)
- [Contoh kebijakan berbasis identitas untuk MemoryDB](#)
- [Memecahkan masalah identitas dan akses MemoryDB](#)
- [Kontrol akses](#)
- [Ikhtisar mengelola izin akses ke sumber daya MemoryDB Anda](#)

Audiens

Bagaimana Anda menggunakan AWS Identity and Access Management (IAM) berbeda, tergantung pada pekerjaan yang Anda lakukan di MemoryDB.

Pengguna layanan - Jika Anda menggunakan layanan MemoryDB untuk melakukan pekerjaan Anda, maka administrator Anda memberi Anda kredensial dan izin yang Anda butuhkan. Saat Anda menggunakan lebih banyak fitur MemoryDB untuk melakukan pekerjaan Anda, Anda mungkin memerlukan izin tambahan. Memahami cara akses dikelola dapat membantu Anda meminta izin

yang tepat dari administrator Anda. Jika Anda tidak dapat mengakses fitur di MemoryDB, lihat.

[Memecahkan masalah identitas dan akses MemoryDB](#)

Administrator layanan - Jika Anda bertanggung jawab atas sumber daya MemoryDB di perusahaan Anda, Anda mungkin memiliki akses penuh ke MemoryDB. Tugas Anda adalah menentukan fitur dan sumber daya MemoryDB mana yang harus diakses pengguna layanan Anda. Kemudian, Anda harus mengirimkan permintaan kepada administrator IAM untuk mengubah izin pengguna layanan Anda. Tinjau informasi di halaman ini untuk memahami konsep Basic IAM. Untuk mempelajari lebih lanjut tentang bagaimana perusahaan Anda dapat menggunakan IAM dengan MemoryDB, lihat.

[Bagaimana MemoryDB bekerja dengan IAM](#)

Administrator IAM - Jika Anda seorang administrator IAM, Anda mungkin ingin mempelajari detail tentang bagaimana Anda dapat menulis kebijakan untuk mengelola akses ke MemoryDB. Untuk melihat contoh kebijakan berbasis identitas MemoryDB yang dapat Anda gunakan di IAM, lihat.

[Contoh kebijakan berbasis identitas untuk MemoryDB](#)

Mengautentikasi dengan identitas

Otentikasi adalah cara Anda masuk AWS menggunakan kredensial identitas Anda. Anda harus diautentikasi (masuk ke AWS) sebagai Pengguna root akun AWS, sebagai pengguna IAM, atau dengan mengasumsikan peran IAM.

Anda dapat masuk AWS sebagai identitas federasi dengan menggunakan kredensial yang disediakan melalui sumber identitas. AWS IAM Identity Center Pengguna (IAM Identity Center), autentikasi masuk tunggal perusahaan Anda, dan kredensial Google atau Facebook Anda adalah contoh identitas federasi. Saat Anda masuk sebagai identitas terfederasi, administrator Anda sebelumnya menyiapkan federasi identitas menggunakan peran IAM. Ketika Anda mengakses AWS dengan menggunakan federasi, Anda secara tidak langsung mengambil peran.

Bergantung pada jenis pengguna Anda, Anda dapat masuk ke AWS Management Console atau portal AWS akses. Untuk informasi selengkapnya tentang masuk AWS, lihat [Cara masuk ke Panduan AWS Sign-In Pengguna Anda Akun AWS](#).

Jika Anda mengakses AWS secara terprogram, AWS sediakan kit pengembangan perangkat lunak (SDK) dan antarmuka baris perintah (CLI) untuk menandatangani permintaan Anda secara kriptografis dengan menggunakan kredensial Anda. Jika Anda tidak menggunakan AWS alat, Anda harus menandatangani permintaan sendiri. Untuk informasi selengkapnya tentang penggunaan metode yang disarankan untuk menandatangani permintaan sendiri, lihat [Menandatangani permintaan AWS API](#) di Panduan Pengguna IAM.

Apa pun metode autentikasi yang digunakan, Anda mungkin diminta untuk menyediakan informasi keamanan tambahan. Misalnya, AWS merekomendasikan agar Anda menggunakan otentikasi multi-faktor (MFA) untuk meningkatkan keamanan akun Anda. Untuk mempelajari selengkapnya, lihat [Autentikasi multi-faktor](#) dalam Panduan Pengguna AWS IAM Identity Center dan [Menggunakan autentikasi multi-faktor \(MFA\) dalam AWS](#) dalam Panduan Pengguna IAM.

Akun AWS pengguna root

Saat Anda membuat Akun AWS, Anda mulai dengan satu identitas masuk yang memiliki akses lengkap ke semua AWS layanan dan sumber daya di akun. Identitas ini disebut pengguna Akun AWS root dan diakses dengan masuk dengan alamat email dan kata sandi yang Anda gunakan untuk membuat akun. Kami sangat menyarankan agar Anda tidak menggunakan pengguna root untuk tugas sehari-hari. Lindungi kredensial pengguna root Anda dan gunakan kredensial tersebut untuk melakukan tugas yang hanya dapat dilakukan pengguna root. Untuk daftar lengkap tugas yang mengharuskan Anda masuk sebagai pengguna root, lihat [Tugas yang memerlukan kredensial pengguna root](#) dalam Panduan Pengguna IAM.

Identitas gabungan

Sebagai praktik terbaik, mewajibkan pengguna manusia, termasuk pengguna yang memerlukan akses administrator, untuk menggunakan federasi dengan penyedia identitas untuk mengakses AWS layanan dengan menggunakan kredensial sementara.

Identitas federasi adalah pengguna dari direktori pengguna perusahaan Anda, penyedia identitas web, direktori Pusat Identitas AWS Directory Service, atau pengguna mana pun yang mengakses AWS layanan dengan menggunakan kredensial yang disediakan melalui sumber identitas. Ketika identitas federasi mengakses Akun AWS, mereka mengambil peran, dan peran memberikan kredensial sementara.

Untuk manajemen akses terpusat, kami sarankan Anda menggunakan AWS IAM Identity Center. Anda dapat membuat pengguna dan grup di Pusat Identitas IAM, atau Anda dapat menghubungkan dan menyinkronkan ke sekumpulan pengguna dan grup di sumber identitas Anda sendiri untuk digunakan di semua aplikasi Akun AWS dan aplikasi Anda. Untuk informasi tentang Pusat Identitas IAM, lihat [Apakah itu Pusat Identitas IAM?](#) dalam Panduan Pengguna AWS IAM Identity Center .

Pengguna dan grup IAM

[Pengguna IAM](#) adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus untuk satu orang atau aplikasi. Jika memungkinkan, kami merekomendasikan untuk mengandalkan kredensial

sementara, bukan membuat pengguna IAM yang memiliki kredensial jangka panjang seperti kata sandi dan kunci akses. Namun, jika Anda memiliki kasus penggunaan tertentu yang memerlukan kredensial jangka panjang dengan pengguna IAM, kami merekomendasikan Anda merotasi kunci akses. Untuk informasi selengkapnya, lihat [Merotasi kunci akses secara teratur untuk kasus penggunaan yang memerlukan kredensial jangka panjang](#) dalam Panduan Pengguna IAM.

[Grup IAM](#) adalah identitas yang menentukan sekumpulan pengguna IAM. Anda tidak dapat masuk sebagai grup. Anda dapat menggunakan grup untuk menentukan izin bagi beberapa pengguna sekaligus. Grup mempermudah manajemen izin untuk sejumlah besar pengguna sekaligus. Misalnya, Anda dapat memiliki grup yang bernama IAMAdmins dan memberikan izin ke grup tersebut untuk mengelola sumber daya IAM.

Pengguna berbeda dari peran. Pengguna secara unik terkait dengan satu orang atau aplikasi, tetapi peran dimaksudkan untuk dapat digunakan oleh siapa pun yang membutuhkannya. Pengguna memiliki kredensial jangka panjang permanen, tetapi peran memberikan kredensial sementara. Untuk mempelajari selengkapnya, lihat [Kapan harus membuat pengguna IAM \(bukan peran\)](#) dalam Panduan Pengguna IAM.

Peran IAM

[Peran IAM](#) adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus. Peran ini mirip dengan pengguna IAM, tetapi tidak terkait dengan orang tertentu. Anda dapat mengambil peran IAM untuk sementara AWS Management Console dengan [beralih peran](#). Anda dapat mengambil peran dengan memanggil operasi AWS CLI atau AWS API atau dengan menggunakan URL kustom. Untuk informasi selengkapnya tentang cara menggunakan peran, lihat [Menggunakan peran IAM](#) dalam Panduan Pengguna IAM.

Peran IAM dengan kredensial sementara berguna dalam situasi berikut:

- Akses pengguna terfederasi – Untuk menetapkan izin ke identitas terfederasi, Anda membuat peran dan menentukan izin untuk peran tersebut. Ketika identitas terfederasi mengautentikasi, identitas tersebut terhubung dengan peran dan diberi izin yang ditentukan oleh peran. Untuk informasi tentang peran untuk federasi, lihat [Membuat peran untuk Penyedia Identitas pihak ketiga](#) dalam Panduan Pengguna IAM. Jika menggunakan Pusat Identitas IAM, Anda harus mengonfigurasi set izin. Untuk mengontrol apa yang dapat diakses identitas Anda setelah identitas tersebut diautentikasi, Pusat Identitas IAM akan mengorelasikan set izin ke peran dalam IAM. Untuk informasi tentang set izin, lihat [Set izin](#) dalam Panduan Pengguna AWS IAM Identity Center .
- Izin pengguna IAM sementara – Pengguna atau peran IAM dapat mengambil peran IAM guna mendapatkan berbagai izin secara sementara untuk tugas tertentu.

- Akses lintas akun – Anda dapat menggunakan peran IAM untuk mengizinkan seseorang (prinsipal tepercaya) di akun lain untuk mengakses sumber daya di akun Anda. Peran adalah cara utama untuk memberikan akses lintas akun. Namun, dengan beberapa AWS layanan, Anda dapat melampirkan kebijakan secara langsung ke sumber daya (alih-alih menggunakan peran sebagai proxy). Untuk mempelajari perbedaan antara peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat [Akses sumber daya lintas akun di IAM di Panduan Pengguna IAM](#).
- Akses lintas layanan — Beberapa AWS layanan menggunakan fitur lain AWS layanan. Sebagai contoh, ketika Anda memanggil suatu layanan, biasanya layanan tersebut menjalankan aplikasi di Amazon EC2 atau menyimpan objek di Amazon S3. Sebuah layanan mungkin melakukannya menggunakan izin prinsipal yang memanggil, menggunakan peran layanan, atau peran terkait layanan.
- Sesi akses teruskan (FAS) — Saat Anda menggunakan pengguna atau peran IAM untuk melakukan tindakan AWS, Anda dianggap sebagai prinsipal. Ketika Anda menggunakan beberapa layanan, Anda mungkin melakukan sebuah tindakan yang kemudian menginisiasi tindakan lain di layanan yang berbeda. FAS menggunakan izin dari pemanggilan utama AWS layanan, dikombinasikan dengan permintaan AWS layanan untuk membuat permintaan ke layanan hilir. Permintaan FAS hanya dibuat ketika layanan menerima permintaan yang memerlukan interaksi dengan orang lain AWS layanan atau sumber daya untuk menyelesaikannya. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan ketika mengajukan permintaan FAS, lihat [Sesi akses maju](#).
- Peran layanan – Peran layanan adalah [peran IAM](#) yang dijalankan oleh layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, mengubah, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat [Membuat sebuah peran untuk mendelegasikan izin ke AWS layanan](#) dalam Panduan pengguna IAM.
- Peran terkait layanan — Peran terkait layanan adalah jenis peran layanan yang ditautkan ke peran layanan. AWS layanan Layanan tersebut dapat menjalankan peran untuk melakukan tindakan atas nama Anda. Peran terkait layanan muncul di Anda Akun AWS dan dimiliki oleh layanan. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.
- Aplikasi yang berjalan di Amazon EC2 — Anda dapat menggunakan peran IAM untuk mengelola kredensi sementara untuk aplikasi yang berjalan pada instans EC2 dan membuat atau permintaan API. AWS CLI AWS Cara ini lebih dianjurkan daripada menyimpan kunci akses dalam instans EC2. Untuk menetapkan AWS peran ke instans EC2 dan membuatnya tersedia untuk semua aplikasinya, Anda membuat profil instance yang dilampirkan ke instance. Profil instans berisi peran dan memungkinkan program yang berjalan di instans EC2 mendapatkan kredensial sementara.

Untuk informasi selengkapnya, lihat [Menggunakan peran IAM untuk memberikan izin ke aplikasi yang berjalan dalam instans Amazon EC2](#) dalam Panduan Pengguna IAM.

Untuk mempelajari apakah kita harus menggunakan peran IAM atau pengguna IAM, lihat [Kapan harus membuat peran IAM \(bukan pengguna\)](#) dalam Panduan Pengguna IAM.

Mengelola akses menggunakan kebijakan

Anda mengontrol akses AWS dengan membuat kebijakan dan melampirkannya ke AWS identitas atau sumber daya. Kebijakan adalah objek AWS yang, ketika dikaitkan dengan identitas atau sumber daya, menentukan izinnya. AWS mengevaluasi kebijakan ini ketika prinsipal (pengguna, pengguna root, atau sesi peran) membuat permintaan. Izin dalam kebijakan menentukan apakah permintaan diizinkan atau ditolak. Sebagian besar kebijakan disimpan AWS sebagai dokumen JSON. Untuk informasi selengkapnya tentang struktur dan isi dokumen kebijakan JSON, lihat [Gambaran umum kebijakan JSON](#) dalam Panduan Pengguna IAM.

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Artinya, prinsipal manakah yang dapat melakukan tindakan pada sumber daya apa, dan dengan kondisi apa.

Secara default, pengguna dan peran tidak memiliki izin. Untuk memberikan izin kepada pengguna untuk melakukan tindakan di sumber daya yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM. Administrator kemudian dapat menambahkan kebijakan IAM ke peran, dan pengguna dapat mengambil peran.

Kebijakan IAM mendefinisikan izin untuk suatu tindakan terlepas dari metode yang Anda gunakan untuk melakukan operasinya. Misalnya, anggaplah Anda memiliki kebijakan yang mengizinkan tindakan `iam:GetRole`. Pengguna dengan kebijakan tersebut bisa mendapatkan informasi peran dari AWS Management Console, API AWS CLI, atau AWS API.

Kebijakan berbasis identitas

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke sebuah identitas, seperti pengguna IAM, grup pengguna IAM, atau peran IAM. Kebijakan ini mengontrol jenis tindakan yang dapat dilakukan oleh pengguna dan peran, di sumber daya mana, dan berdasarkan kondisi seperti apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat [Membuat kebijakan IAM](#) dalam Panduan Pengguna IAM.

Kebijakan berbasis identitas dapat dikategorikan lebih lanjut sebagai kebijakan inline atau kebijakan yang dikelola. Kebijakan inline disematkan langsung ke satu pengguna, grup, atau peran. Kebijakan terkelola adalah kebijakan mandiri yang dapat Anda lampirkan ke beberapa pengguna, grup, dan peran dalam. Akun AWS Kebijakan AWS terkelola mencakup kebijakan terkelola dan kebijakan yang dikelola pelanggan. Untuk mempelajari cara memilih antara kebijakan yang dikelola atau kebijakan inline, lihat [Memilih antara kebijakan yang dikelola dan kebijakan inline](#) dalam Panduan Pengguna IAM.

Kebijakan berbasis sumber daya

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan kepercayaan peran IAM dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Untuk sumber daya tempat kebijakan dilampirkan, kebijakan menentukan tindakan apa yang dapat dilakukan oleh prinsipal tertentu pada sumber daya tersebut dan dalam kondisi apa. Anda harus [menentukan prinsipal](#) dalam kebijakan berbasis sumber daya. Prinsipal dapat mencakup akun, pengguna, peran, pengguna federasi, atau. AWS layanan

Kebijakan berbasis sumber daya merupakan kebijakan inline yang terletak di layanan tersebut. Anda tidak dapat menggunakan kebijakan AWS terkelola dari IAM dalam kebijakan berbasis sumber daya.

Daftar kontrol akses (ACL)

Daftar kontrol akses (ACL) mengendalikan prinsipal mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACL serupa dengan kebijakan berbasis sumber daya, meskipun kebijakan tersebut tidak menggunakan format dokumen kebijakan JSON.

Amazon S3, AWS WAF, dan Amazon VPC adalah contoh layanan yang mendukung ACL. Untuk mempelajari ACL selengkapnya, lihat [Gambaran umum daftar kontrol akses \(ACL\)](#) dalam Panduan Developer Amazon Simple Storage Service.

Jenis-jenis kebijakan lain

AWS mendukung jenis kebijakan tambahan yang kurang umum. Jenis-jenis kebijakan ini dapat mengatur izin maksimum yang diberikan kepada Anda oleh jenis kebijakan yang lebih umum.

- Batasan izin – Batasan izin adalah fitur lanjutan tempat Anda mengatur izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas ke entitas IAM (pengguna IAM atau peran IAM). Anda dapat menetapkan batasan izin untuk suatu entitas. Izin yang dihasilkan adalah perpotongan

antara kebijakan berbasis identitas milik entitas dan batasan izinnya. Kebijakan berbasis sumber daya yang menentukan pengguna atau peran dalam bidang `Principal` tidak dibatasi oleh batasan izin. Penolakan eksplisit dalam salah satu kebijakan ini akan menggantikan pemberian izin. Untuk informasi selengkapnya tentang batasan izin, lihat [Batasan izin untuk entitas IAM](#) dalam Panduan Pengguna IAM.

- Kebijakan kontrol layanan (SCP) — SCP adalah kebijakan JSON yang menentukan izin maksimum untuk organisasi atau unit organisasi (OU) di AWS Organizations. AWS Organizations adalah layanan untuk mengelompokkan dan mengelola secara terpusat beberapa Akun AWS yang dimiliki bisnis Anda. Jika Anda mengaktifkan semua fitur di organisasi, Anda dapat menerapkan kebijakan kontrol layanan (SCP) ke salah satu atau semua akun Anda. SCP membatasi izin untuk entitas di akun anggota, termasuk masing-masing. Pengguna root akun AWS Untuk informasi selengkapnya tentang Organisasi dan SCP, lihat [Cara kerja SCP](#) dalam Panduan Pengguna AWS Organizations.
- Kebijakan sesi – Kebijakan sesi adalah kebijakan lanjutan yang Anda berikan sebagai parameter ketika Anda membuat sesi sementara secara programatis untuk peran atau pengguna terfederasi. Izin sesi yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas pengguna atau peran dan kebijakan sesi. Izin juga bisa datang dari kebijakan berbasis sumber daya. Penolakan eksplisit dalam salah satu kebijakan ini akan menggantikan pemberian izin. Untuk informasi selengkapnya, lihat [Kebijakan sesi](#) dalam Panduan Pengguna IAM.

Berbagai jenis kebijakan

Ketika beberapa jenis kebijakan berlaku pada suatu permintaan, izin yang dihasilkan lebih rumit untuk dipahami. Untuk mempelajari cara AWS menentukan apakah akan mengizinkan permintaan saat beberapa jenis kebijakan terlibat, lihat [Logika evaluasi kebijakan](#) di Panduan Pengguna IAM.

Bagaimana MemoryDB bekerja dengan IAM

Sebelum Anda menggunakan IAM untuk mengelola akses ke MemoryDB, pelajari fitur IAM apa yang tersedia untuk digunakan dengan MemoryDB.

Fitur IAM yang dapat Anda gunakan dengan MemoryDB

Fitur IAM	Dukungan MemoryDB
Kebijakan berbasis identitas	Ya
Kebijakan berbasis sumber daya	Tidak

Fitur IAM	Dukungan MemoryDB
Tindakan kebijakan	Ya
Sumber daya kebijakan	Ya
Kunci kondisi kebijakan	Ya
ACL	Ya
ABAC (tanda dalam kebijakan)	Ya
Kredensial sementara	Ya
Izin prinsipal	Ya
Peran layanan	Ya
Peran terkait layanan	Ya

Untuk mendapatkan tampilan tingkat tinggi tentang cara kerja MemoryDB dan AWS layanan lainnya dengan sebagian besar fitur IAM, lihat [AWS layanan yang bekerja dengan IAM di Panduan Pengguna IAM](#).

Kebijakan berbasis identitas untuk MemoryDB

Mendukung kebijakan berbasis identitas: Ya

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke sebuah identitas, seperti pengguna IAM, grup pengguna IAM, atau peran IAM. Kebijakan ini mengontrol jenis tindakan yang dapat dilakukan oleh pengguna dan peran, di sumber daya mana, dan berdasarkan kondisi seperti apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat [Membuat kebijakan IAM](#) dalam Panduan Pengguna IAM.

Dengan kebijakan berbasis identitas IAM, Anda dapat menentukan secara spesifik apakah tindakan dan sumber daya diizinkan atau ditolak, serta kondisi yang menjadi dasar dikabulkan atau ditolaknya tindakan tersebut. Anda tidak dapat menentukan secara spesifik prinsipal dalam sebuah kebijakan berbasis identitas karena prinsipal berlaku bagi pengguna atau peran yang melekat kepadanya. Untuk mempelajari semua elemen yang dapat Anda gunakan dalam kebijakan JSON, lihat [Referensi elemen kebijakan JSON IAM](#) dalam Panduan Pengguna IAM.

Contoh kebijakan berbasis identitas untuk MemoryDB

Untuk melihat contoh kebijakan berbasis identitas MemoryDB, lihat. [Contoh kebijakan berbasis identitas untuk MemoryDB](#)

Kebijakan berbasis sumber daya dalam MemoryDB

Mendukung kebijakan berbasis sumber daya: Tidak

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan kepercayaan peran IAM dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Untuk sumber daya tempat kebijakan dilampirkan, kebijakan menentukan tindakan apa yang dapat dilakukan oleh prinsipal tertentu pada sumber daya tersebut dan dalam kondisi apa. Anda harus [menentukan prinsipal](#) dalam kebijakan berbasis sumber daya. Prinsipal dapat mencakup akun, pengguna, peran, pengguna federasi, atau AWS layanan

Untuk mengaktifkan akses lintas akun, Anda dapat menentukan secara spesifik seluruh akun atau entitas IAM di akun lain sebagai prinsipal dalam kebijakan berbasis sumber daya. Menambahkan prinsipal akun silang ke kebijakan berbasis sumber daya hanya setengah dari membangun hubungan kepercayaan. Ketika prinsipal dan sumber daya berbeda Akun AWS, administrator IAM di akun tepercaya juga harus memberikan izin entitas utama (pengguna atau peran) untuk mengakses sumber daya. Mereka memberikan izin dengan melampirkan kebijakan berbasis identitas kepada entitas. Namun, jika kebijakan berbasis sumber daya memberikan akses ke prinsipal dalam akun yang sama, tidak diperlukan kebijakan berbasis identitas tambahan. Untuk informasi selengkapnya, lihat [Akses sumber daya lintas akun di IAM](#) di Panduan Pengguna IAM.

Tindakan kebijakan untuk MemoryDB

Mendukung tindakan kebijakan: Ya

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Artinya, prinsipal manakah yang dapat melakukan tindakan pada sumber daya apa, dan dengan kondisi apa.

Elemen `Action` dari kebijakan JSON menjelaskan tindakan yang dapat Anda gunakan untuk mengizinkan atau menolak akses dalam sebuah kebijakan. Tindakan kebijakan biasanya memiliki nama yang sama dengan operasi AWS API terkait. Ada beberapa pengecualian, misalnya tindakan

hanya izin yang tidak memiliki operasi API yang cocok. Ada juga beberapa operasi yang memerlukan beberapa tindakan dalam suatu kebijakan. Tindakan tambahan ini disebut tindakan dependen.

Menyertakan tindakan dalam kebijakan untuk memberikan izin untuk melakukan operasi terkait.

Untuk melihat daftar tindakan MemoryDB, lihat [Tindakan yang Ditetapkan oleh MemoryDB](#) di Referensi Otorisasi Layanan.

Tindakan kebijakan di MemoryDB menggunakan awalan berikut sebelum tindakan:

```
MemoryDB
```

Untuk menetapkan secara spesifik beberapa tindakan dalam satu pernyataan, pisahkan tindakan tersebut dengan koma.

```
"Action": [  
  "MemoryDB:action1",  
  "MemoryDB:action2"  
]
```

Anda juga dapat menentukan beberapa tindakan menggunakan wildcard (*). Sebagai contoh, untuk menentukan semua tindakan yang dimulai dengan kata `Describe`, sertakan tindakan berikut:

```
"Action": "MemoryDB:Describe*"
```

Untuk melihat contoh kebijakan berbasis identitas MemoryDB, lihat [Contoh kebijakan berbasis identitas untuk MemoryDB](#)

Sumber daya kebijakan untuk MemoryDB

Mendukung sumber daya kebijakan: Ya

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Artinya, prinsipal manakah yang dapat melakukan tindakan pada sumber daya apa, dan dengan kondisi apa.

Elemen kebijakan JSON `Resource` menentukan objek yang menjadi target penerapan tindakan. Pernyataan harus menyertakan elemen `Resource` atau `NotResource`. Praktik terbaiknya, tentukan

sumber daya menggunakan [Amazon Resource Name \(ARN\)](#). Anda dapat melakukan ini untuk tindakan yang mendukung jenis sumber daya tertentu, yang dikenal sebagai izin tingkat sumber daya.

Untuk tindakan yang tidak mendukung izin di tingkat sumber daya, misalnya operasi pencantuman, gunakan wildcard (*) untuk menunjukkan bahwa pernyataan tersebut berlaku untuk semua sumber daya.

```
"Resource": "*"
```

Untuk melihat daftar jenis sumber daya MemoryDB dan ARNnya, lihat Sumber Daya yang [Ditentukan oleh MemoryDB](#) di Referensi Otorisasi Layanan. Untuk mempelajari tindakan mana yang dapat Anda tentukan ARN dari setiap sumber daya, lihat [Tindakan yang Ditentukan oleh](#) MemoryDB.

Untuk melihat contoh kebijakan berbasis identitas MemoryDB, lihat. [Contoh kebijakan berbasis identitas untuk MemoryDB](#)

Kunci kondisi kebijakan untuk MemoryDB

Mendukung kunci kondisi kebijakan khusus layanan: Ya

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Artinya, prinsipal manakah yang dapat melakukan tindakan pada sumber daya apa, dan dengan kondisi apa.

Elemen `Condition` (atau blok `Condition`) akan memungkinkan Anda menentukan kondisi yang menjadi dasar suatu pernyataan berlaku. Elemen `Condition` bersifat opsional. Anda dapat membuat ekspresi bersyarat yang menggunakan [operator kondisi](#), misalnya sama dengan atau kurang dari, untuk mencocokkan kondisi dalam kebijakan dengan nilai-nilai yang diminta.

Jika Anda menentukan beberapa elemen `Condition` dalam sebuah pernyataan, atau beberapa kunci dalam elemen `Condition` tunggal, maka AWS akan mengevaluasinya menggunakan operasi AND logis. Jika Anda menentukan beberapa nilai untuk satu kunci kondisi, AWS mengevaluasi kondisi menggunakan OR operasi logis. Semua kondisi harus dipenuhi sebelum izin pernyataan diberikan.

Anda juga dapat menggunakan variabel placeholder saat menentukan kondisi. Sebagai contoh, Anda dapat memberikan izin kepada pengguna IAM untuk mengakses sumber daya hanya jika izin tersebut mempunyai tag yang sesuai dengan nama pengguna IAM mereka. Untuk informasi selengkapnya, lihat [Elemen kebijakan IAM: variabel dan tag](#) dalam Panduan Pengguna IAM.

AWS mendukung kunci kondisi global dan kunci kondisi khusus layanan. Untuk melihat semua kunci kondisi AWS global, lihat [kunci konteks kondisi AWS global](#) di Panduan Pengguna IAM.

Untuk melihat contoh kebijakan berbasis identitas MemoryDB, lihat [Contoh kebijakan berbasis identitas untuk MemoryDB](#)

Menggunakan kunci kondisi

Anda dapat menentukan kondisi yang menentukan cara kebijakan IAM diberlakukan. Di MemoryDB, Anda dapat menggunakan `Condition` elemen kebijakan JSON untuk membandingkan kunci dalam konteks permintaan dengan nilai kunci yang Anda tentukan dalam kebijakan Anda. Untuk informasi selengkapnya, lihat [Elemen kebijakan JSON IAM: Kondisi](#).

Untuk melihat daftar kunci kondisi MemoryDB, lihat Kunci Kondisi [untuk MemoryDB](#) di Referensi Otorisasi Layanan.

Untuk melihat daftar kunci kondisi global, lihat [Kunci konteks kondisi global AWS](#).

Menentukan Kondisi: Menggunakan Kunci Kondisi

Untuk menerapkan kontrol berbutir halus, Anda dapat menulis kebijakan izin IAM yang menentukan kondisi untuk mengontrol sekumpulan parameter individual pada permintaan tertentu. Anda kemudian dapat menerapkan kebijakan tersebut ke pengguna, grup, atau peran IAM yang Anda buat menggunakan konsol IAM.

Untuk menerapkan kondisi tersebut, Anda menambahkan informasi kondisi pada pernyataan kebijakan IAM. Misalnya, untuk melarang pembuatan kluster MemoryDB dengan TLS dinonaktifkan, Anda dapat menentukan kondisi berikut dalam pernyataan kebijakan Anda.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "memorydb:CreateCluster"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "Bool": {
          "memorydb:TLSEnabled": "false"
        }
      }
    }
  ]
}
```

```

    }
  }
}
]
}

```

Untuk informasi lebih lanjut tentang penandaan, lihat [Menandai sumber daya MemoryDB Anda](#).

Untuk informasi selengkapnya tentang penggunaan operator kondisi kebijakan, lihat [Izin API MemoryDB: Referensi tindakan, sumber daya, dan kondisi](#).

Kebijakan Contoh: Menggunakan Kondisi untuk Kontrol Parameter Terperinci

Bagian ini menunjukkan contoh kebijakan untuk menerapkan kontrol akses berbutir halus pada parameter MemoryDB yang terdaftar sebelumnya.

1. MemoryDB:TLSEnabled - Tentukan bahwa cluster akan dibuat hanya dengan TLS diaktifkan.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "memorydb:CreateCluster"
      ],
      "Resource": [
        "arn:aws:memorydb:*:*:parametergroup/*",
        "arn:aws:memorydb:*:*:subnetgroup/*",
        "arn:aws:memorydb:*:*:acl/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "memorydb:CreateCluster"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "Bool": {
          "memorydb:TLSEnabled": "true"
        }
      }
    }
  ]
}

```

```

    }
  }
]
}

```

2. `memorydb:UserAuthenticationMode`: — Tentukan bahwa pengguna dapat dibuat dengan mode otentikasi tipe tertentu (IAM misalnya).

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "memorydb:Createuser"
      ],
      "Resource": [
        "arn:aws:memorydb:*:*:user/*"
      ],
      "Condition": {
        "StringEquals": {
          "memorydb:UserAuthenticationMode": "iam"
        }
      }
    }
  ]
}

```

Dalam kasus di mana Anda menyetel kebijakan berbasis 'Tolak', disarankan untuk menggunakan [StringEqualsIgnoreCase](#) operator untuk menghindari semua panggilan dengan jenis mode otentikasi pengguna tertentu terlepas dari kasusnya.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "memorydb:CreateUser"
      ],
      "Resource": "*",

```

```
    "Condition": {
      "StringEqualsIgnoreCase": {
        "memorydb:UserAuthenticationMode": "password"
      }
    }
  }
]
```

Daftar kontrol akses (ACL) di MemoryDB

Mendukung ACL: Ya

Daftar kontrol akses (ACL) mengendalikan prinsipal mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACL serupa dengan kebijakan berbasis sumber daya, meskipun kebijakan tersebut tidak menggunakan format dokumen kebijakan JSON.

Kontrol akses berbasis atribut (ABAC) dengan MemoryDB

Mendukung ABAC (tag dalam kebijakan): Ya

Kontrol akses berbasis atribut (ABAC) adalah strategi otorisasi yang menentukan izin berdasarkan atribut. Dalam AWS, atribut ini disebut tag. Anda dapat melampirkan tag ke entitas IAM (pengguna atau peran) dan ke banyak AWS sumber daya. Penandaan ke entitas dan sumber daya adalah langkah pertama dari ABAC. Kemudian rancanglah kebijakan ABAC untuk mengizinkan operasi ketika tag milik prinsipal cocok dengan tag yang ada di sumber daya yang ingin diakses.

ABAC sangat berguna di lingkungan yang berkembang dengan cepat dan berguna di situasi saat manajemen kebijakan menjadi rumit.

Untuk mengendalikan akses berdasarkan tag, berikan informasi tentang tag di [elemen kondisi](#) dari kebijakan menggunakan kunci kondisi `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, atau `aws:TagKeys`.

Jika sebuah layanan mendukung ketiga kunci kondisi untuk setiap jenis sumber daya, nilainya adalah Ya untuk layanan tersebut. Jika suatu layanan mendukung ketiga kunci kondisi untuk hanya beberapa jenis sumber daya, nilainya adalah Parsial.

Untuk informasi selengkapnya tentang ABAC, lihat [Apa itu ABAC?](#) dalam Panduan Pengguna IAM. Untuk melihat tutorial yang menguraikan langkah-langkah pengaturan ABAC, lihat [Menggunakan kontrol akses berbasis atribut \(ABAC\)](#) dalam Panduan Pengguna IAM.

Menggunakan kredensial Sementara dengan MemoryDB

Mendukung kredensial sementara: Ya

Beberapa AWS layanan tidak berfungsi saat Anda masuk menggunakan kredensial sementara. Untuk informasi tambahan, termasuk yang AWS layanan bekerja dengan kredensial sementara, lihat [AWS layanan yang bekerja dengan IAM di Panduan Pengguna IAM](#).

Anda menggunakan kredensial sementara jika Anda masuk AWS Management Console menggunakan metode apa pun kecuali nama pengguna dan kata sandi. Misalnya, ketika Anda mengakses AWS menggunakan tautan masuk tunggal (SSO) perusahaan Anda, proses tersebut secara otomatis membuat kredensial sementara. Anda juga akan secara otomatis membuat kredensial sementara ketika Anda masuk ke konsol sebagai seorang pengguna lalu beralih peran. Untuk informasi selengkapnya tentang peralihan peran, lihat [Peralihan peran \(konsol\)](#) dalam Panduan Pengguna IAM.

Anda dapat membuat kredensial sementara secara manual menggunakan API AWS CLI atau AWS . Anda kemudian dapat menggunakan kredensial sementara tersebut untuk mengakses. AWS AWS merekomendasikan agar Anda secara dinamis menghasilkan kredensial sementara alih-alih menggunakan kunci akses jangka panjang. Untuk informasi selengkapnya, lihat [Kredensial keamanan sementara di IAM](#).

Izin utama lintas layanan untuk MemoryDB

Mendukung sesi akses maju (FAS): Ya

Saat Anda menggunakan pengguna atau peran IAM untuk melakukan tindakan AWS, Anda dianggap sebagai prinsipal. Ketika Anda menggunakan beberapa layanan, Anda mungkin melakukan sebuah tindakan yang kemudian menginisiasi tindakan lain di layanan yang berbeda. FAS menggunakan izin dari pemanggilan utama AWS layanan, dikombinasikan dengan permintaan AWS layanan untuk membuat permintaan ke layanan hilir. Permintaan FAS hanya dibuat ketika layanan menerima permintaan yang memerlukan interaksi dengan orang lain AWS layanan atau sumber daya untuk menyelesaikannya. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan ketika mengajukan permintaan FAS, lihat [Sesi akses maju](#).

Peran layanan untuk MemoryDB

Mendukung peran layanan: Ya

Peran layanan adalah [peran IAM](#) yang diambil oleh sebuah layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, mengubah, dan menghapus peran layanan dari

dalam IAM. Untuk informasi selengkapnya, lihat [Membuat sebuah peran untuk mendelegasikan izin ke AWS layanan](#) dalam Panduan pengguna IAM.

Warning

Mengubah izin untuk peran layanan dapat merusak fungsionalitas MemoryDB. Edit peran layanan hanya ketika MemoryDB memberikan panduan untuk melakukannya.

Peran terkait layanan untuk MemoryDB

Mendukung peran terkait layanan: Ya

Peran terkait layanan adalah jenis peran layanan yang ditautkan ke AWS layanan Layanan tersebut dapat menjalankan peran untuk melakukan tindakan atas nama Anda. Peran terkait layanan muncul di Anda Akun AWS dan dimiliki oleh layanan. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.

Untuk detail tentang pembuatan atau manajemen peran terkait layanan, lihat [Layanan AWS yang berfungsi dengan IAM](#). Cari layanan dalam tabel yang memiliki Yes di kolom Peran terkait layanan. Pilih tautan Ya untuk melihat dokumentasi peran terkait layanan untuk layanan tersebut.

Contoh kebijakan berbasis identitas untuk MemoryDB

Secara default, pengguna dan peran tidak memiliki izin untuk membuat atau memodifikasi sumber daya MemoryDB. Mereka juga tidak dapat melakukan tugas dengan menggunakan AWS Management Console, AWS Command Line Interface (AWS CLI), atau AWS API. Untuk memberikan izin kepada pengguna untuk melakukan tindakan di sumber daya yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM. Administrator kemudian akan dapat menambahkan kebijakan IAM ke peran, dan pengguna dapat mengambil peran.

Untuk mempelajari cara membuat kebijakan berbasis identitas IAM menggunakan contoh dokumen kebijakan JSON ini, lihat [Membuat kebijakan IAM](#) dalam Panduan Pengguna IAM.

Untuk detail tentang tindakan dan jenis sumber daya yang ditentukan oleh MemoryDB, termasuk format ARN untuk setiap jenis sumber daya, lihat [Tindakan, Sumber Daya, dan Kunci Kondisi untuk MemoryDB](#) di Referensi Otorisasi Layanan.

Topik

- [Praktik terbaik kebijakan](#)
- [Menggunakan konsol MemoryDB](#)
- [Mengizinkan pengguna melihat izin mereka sendiri](#)

Praktik terbaik kebijakan

Kebijakan berbasis identitas menentukan apakah seseorang dapat membuat, mengakses, atau menghapus sumber daya MemoryDB di akun Anda. Tindakan ini membuat Akun AWS Anda dikenai biaya. Ketika Anda membuat atau mengedit kebijakan berbasis identitas, ikuti panduan dan rekomendasi ini:

- Mulailah dengan kebijakan AWS terkelola dan beralih ke izin hak istimewa paling sedikit — Untuk mulai memberikan izin kepada pengguna dan beban kerja Anda, gunakan kebijakan AWS terkelola yang memberikan izin untuk banyak kasus penggunaan umum. Mereka tersedia di Anda Akun AWS. Kami menyarankan Anda mengurangi izin lebih lanjut dengan menentukan kebijakan yang dikelola AWS pelanggan yang khusus untuk kasus penggunaan Anda. Untuk informasi selengkapnya, lihat [Kebijakan yang dikelola AWS](#) atau [Kebijakan yang dikelola AWS untuk fungsi tugas](#) dalam Panduan Pengguna IAM.
- Menerapkan izin dengan hak akses paling rendah – Ketika Anda menetapkan izin dengan kebijakan IAM, hanya berikan izin yang diperlukan untuk melakukan tugas. Anda melakukannya dengan mendefinisikan tindakan yang dapat diambil pada sumber daya tertentu dalam kondisi tertentu, yang juga dikenal sebagai izin dengan hak akses paling rendah. Untuk informasi selengkapnya tentang cara menggunakan IAM untuk mengajukan izin, lihat [Kebijakan dan izin dalam IAM](#) dalam Panduan Pengguna IAM.
- Gunakan kondisi dalam kebijakan IAM untuk membatasi akses lebih lanjut – Anda dapat menambahkan suatu kondisi ke kebijakan Anda untuk membatasi akses ke tindakan dan sumber daya. Sebagai contoh, Anda dapat menulis kondisi kebijakan untuk menentukan bahwa semua permintaan harus dikirim menggunakan SSL. Anda juga dapat menggunakan ketentuan untuk memberikan akses ke tindakan layanan jika digunakan melalui yang spesifik AWS layanan, seperti AWS CloudFormation. Untuk informasi selengkapnya, lihat [Elemen kebijakan JSON IAM: Kondisi](#) dalam Panduan Pengguna IAM.
- Gunakan IAM Access Analyzer untuk memvalidasi kebijakan IAM Anda untuk memastikan izin yang aman dan fungsional – IAM Access Analyzer memvalidasi kebijakan baru dan yang sudah ada sehingga kebijakan tersebut mematuhi bahasa kebijakan IAM (JSON) dan praktik terbaik IAM. IAM Access Analyzer menyediakan lebih dari 100 pemeriksaan kebijakan dan rekomendasi yang dapat ditindaklanjuti untuk membantu Anda membuat kebijakan yang aman dan fungsional. Untuk

informasi selengkapnya, lihat [Validasi kebijakan IAM Access Analyzer](#) dalam Panduan Pengguna IAM.

- Memerlukan otentikasi multi-faktor (MFA) - Jika Anda memiliki skenario yang mengharuskan pengguna IAM atau pengguna root di Anda, Akun AWS aktifkan MFA untuk keamanan tambahan. Untuk meminta MFA ketika operasi API dipanggil, tambahkan kondisi MFA pada kebijakan Anda. Untuk informasi selengkapnya, lihat [Mengonfigurasi akses API yang dilindungi MFA](#) dalam Panduan Pengguna IAM.

Untuk informasi selengkapnya tentang praktik terbaik dalam IAM, lihat [Praktik terbaik keamanan dalam IAM](#) dalam Panduan Pengguna IAM.

Menggunakan konsol MemoryDB

Untuk mengakses konsol MemoryDB, Anda harus memiliki set izin minimum. Izin ini harus memungkinkan Anda untuk daftar dan melihat rincian tentang sumber daya MemoryDB di Anda. Akun AWS Jika Anda membuat kebijakan berbasis identitas yang lebih ketat daripada izin minimum yang diperlukan, konsol tidak akan berfungsi sebagaimana mestinya untuk entitas (pengguna atau peran) dengan kebijakan tersebut.

Anda tidak perlu mengizinkan izin konsol minimum untuk pengguna yang melakukan panggilan hanya ke AWS CLI atau AWS API. Sebagai gantinya, izinkan akses hanya ke tindakan yang sesuai dengan operasi API yang coba mereka lakukan.

Untuk memastikan bahwa pengguna dan peran masih dapat menggunakan konsol MemoryDB, lampirkan juga MemoryDB ConsoleAccess atau kebijakan ReadOnly AWS terkelola ke entitas. Untuk informasi selengkapnya, lihat [Menambah izin untuk pengguna](#) dalam Panduan Pengguna IAM.

Mengizinkan pengguna melihat izin mereka sendiri

Contoh ini menunjukkan cara membuat kebijakan yang mengizinkan pengguna IAM melihat kebijakan inline dan terkelola yang dilampirkan ke identitas pengguna mereka. Kebijakan ini mencakup izin untuk menyelesaikan tindakan ini di konsol atau menggunakan API atau secara terprogram. AWS CLI AWS

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```

    "Sid": "ViewOwnUserInfo",
    "Effect": "Allow",
    "Action": [
        "iam:GetUserPolicy",
        "iam:ListGroupsForUser",
        "iam:ListAttachedUserPolicies",
        "iam:ListUserPolicies",
        "iam:GetUser"
    ],
    "Resource": ["arn:aws:iam::*:user/${aws:username}"]
},
{
    "Sid": "NavigateInConsole",
    "Effect": "Allow",
    "Action": [
        "iam:GetGroupPolicy",
        "iam:GetPolicyVersion",
        "iam:GetPolicy",
        "iam:ListAttachedGroupPolicies",
        "iam:ListGroupPolicies",
        "iam:ListPolicyVersions",
        "iam:ListPolicies",
        "iam:ListUsers"
    ],
    "Resource": "*"
}
]
}

```

Memecahkan masalah identitas dan akses MemoryDB

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah umum yang mungkin Anda temui saat bekerja dengan MemoryDB dan IAM.

Topik

- [Saya tidak berwenang untuk melakukan tindakan di MemoryDB](#)
- [Saya tidak berwenang untuk melakukan iam: PassRole](#)
- [Saya ingin mengizinkan orang di luar AWS akun saya untuk mengakses sumber daya MemoryDB saya](#)

Saya tidak berwenang untuk melakukan tindakan di MemoryDB

Jika AWS Management Console memberitahu Anda bahwa Anda tidak berwenang untuk melakukan tindakan, maka Anda harus menghubungi administrator Anda untuk bantuan. Administrator Anda adalah orang yang memberikan nama pengguna dan kata sandi Anda.

Contoh kesalahan berikut terjadi ketika pengguna `mateojackson` mencoba menggunakan konsol untuk melihat detail tentang suatu sumber daya `my-example-widget` fiktif, tetapi tidak memiliki izin MemoryDB: `GetWidget` fiktif.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
MemoryDB:GetWidget on resource: my-example-widget
```

Dalam hal ini, Mateo meminta administratornya untuk memperbarui kebijakannya untuk mengizinkan dia mengakses sumber daya `my-example-widget` menggunakan tindakan MemoryDB: `GetWidget`.

Saya tidak berwenang untuk melakukan iam: PassRole

Jika Anda menerima kesalahan bahwa Anda tidak berwenang untuk melakukan `iam:PassRole` tindakan, kebijakan Anda harus diperbarui agar Anda dapat meneruskan peran ke MemoryDB.

Beberapa AWS layanan memungkinkan Anda untuk meneruskan peran yang ada ke layanan tersebut alih-alih membuat peran layanan baru atau peran terkait layanan. Untuk melakukannya, Anda harus memiliki izin untuk meneruskan peran ke layanan.

Contoh kesalahan berikut terjadi ketika pengguna IAM bernama `marymajor` mencoba menggunakan konsol untuk melakukan tindakan di MemoryDB. Namun, tindakan tersebut memerlukan layanan untuk mendapatkan izin yang diberikan oleh peran layanan. Mary tidak memiliki izin untuk meneruskan peran tersebut pada layanan.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

Dalam kasus ini, kebijakan Mary harus diperbarui agar dia mendapatkan izin untuk melakukan tindakan `iam:PassRole` tersebut.

Jika Anda memerlukan bantuan, hubungi AWS administrator Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Saya ingin mengizinkan orang di luar AWS akun saya untuk mengakses sumber daya MemoryDB saya

Anda dapat membuat peran yang dapat digunakan pengguna di akun lain atau orang-orang di luar organisasi Anda untuk mengakses sumber daya Anda. Anda dapat menentukan siapa saja yang dipercaya untuk mengambil peran tersebut. Untuk layanan yang mendukung kebijakan berbasis sumber daya atau daftar kontrol akses (ACL), Anda dapat menggunakan kebijakan tersebut untuk memberi orang akses ke sumber daya Anda.

Untuk mempelajari selengkapnya, periksa referensi berikut:

- Untuk mempelajari apakah MemoryDB mendukung fitur-fitur ini, lihat [Bagaimana MemoryDB bekerja dengan IAM](#)
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda di seluruh sumber daya Akun AWS yang Anda miliki, lihat [Menyediakan akses ke pengguna IAM di pengguna lain Akun AWS yang Anda miliki](#) di Panduan Pengguna IAM.
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda kepada pihak ketiga Akun AWS, lihat [Menyediakan akses yang Akun AWS dimiliki oleh pihak ketiga](#) dalam Panduan Pengguna IAM.
- Untuk mempelajari cara memberikan akses melalui federasi identitas, lihat [Menyediakan akses ke pengguna terautentikasi eksternal \(federasi identitas\)](#) dalam Panduan Pengguna IAM.
- Untuk mempelajari perbedaan antara menggunakan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat [Akses sumber daya lintas akun di IAM di Panduan Pengguna IAM](#).

Kontrol akses

Anda dapat memiliki kredensial yang valid untuk mengautentikasi permintaan Anda, tetapi kecuali Anda memiliki izin, Anda tidak dapat membuat atau mengakses sumber daya MemoryDB. Misalnya, Anda harus memiliki izin untuk membuat cluster MemoryDB.

Bagian berikut menjelaskan cara mengelola izin untuk MemoryDB. Anda sebaiknya membaca gambaran umum terlebih dahulu.

- [Ikhtisar mengelola izin akses ke sumber daya MemoryDB Anda](#)
- [Menggunakan kebijakan berbasis identitas \(kebijakan IAM\) untuk MemoryDB](#)

Ikhtisar mengelola izin akses ke sumber daya MemoryDB Anda

Setiap AWS sumber daya dimiliki oleh AWS akun, dan izin untuk membuat atau mengakses sumber daya diatur oleh kebijakan izin. Administrator akun dapat melampirkan kebijakan izin pada identitas IAM (yaitu pengguna, grup, dan peran). Selain itu, MemoryDB juga mendukung melampirkan kebijakan izin ke sumber daya.

Note

Administrator akun (atau pengguna administrator) adalah pengguna dengan hak akses administrator. Untuk informasi selengkapnya, lihat [Praktik Terbaik IAM](#) dalam Panduan Pengguna IAM.

Untuk memberikan akses, tambahkan izin ke pengguna, grup, atau peran Anda:

- Pengguna dan grup di AWS IAM Identity Center:

Buat rangkaian izin. Ikuti instruksi di [Buat rangkaian izin](#) di Panduan Pengguna AWS IAM Identity Center .

- Pengguna yang dikelola di IAM melalui penyedia identitas:

Buat peran untuk federasi identitas. Ikuti instruksi dalam [Membuat peran untuk penyedia identitas pihak ketiga \(federasi\)](#) di Panduan Pengguna IAM.

- Pengguna IAM:

- Buat peran yang dapat diambil pengguna Anda. Ikuti instruksi dalam [Membuat peran untuk pengguna IAM](#) dalam Panduan Pengguna IAM.
- (Tidak disarankan) Pasang kebijakan langsung ke pengguna atau tambahkan pengguna ke grup pengguna. Ikuti instruksi dalam [Menambahkan izin ke pengguna \(konsol\)](#) dalam Panduan Pengguna IAM.

Topik

- [Sumber daya dan operasi MemoryDB](#)
- [Memahami kepemilikan sumber daya](#)
- [Mengelola akses ke sumber daya](#)
- [Menggunakan kebijakan berbasis identitas \(kebijakan IAM\) untuk MemoryDB](#)

- [Izin tingkat sumber daya](#)
- [Menggunakan Peran Tertaut Layanan untuk MemoryDB](#)
- [AWS kebijakan terkelola untuk MemoryDB](#)
- [Izin API MemoryDB: Referensi tindakan, sumber daya, dan kondisi](#)

Sumber daya dan operasi MemoryDB

Di MemoryDB, sumber daya utama adalah cluster.

Sumber daya ini memiliki Amazon Resource Name (ARN) yang unik dan terkait dengannya, seperti ditunjukkan berikut ini.

Note

Agar izin di tingkat sumber daya menjadi efektif, nama sumber daya pada string ARN harus huruf kecil.

Jenis sumber daya	Format ARN
Pengguna	<code>arn:aws:memorydb: us-timur-1:123456789012: pengguna/pengguna1</code>
Daftar Kontrol Akses (ACL)	<code>arn:aws:memorydb: us-timur-1:123456789012: acl/myacl</code>
Klaster	<code>arn:aws:memorydb: us-timur-1:123456789012: cluster/my-cluster</code>
Snapshot	<code>arn:aws:memorydb: us-timur-1:123456789012: snapshot/snapshot saya</code>

Jenis sumber daya	Format ARN
Grup parameter	<i>arn:aws:memorydb: us-timur-1:123456789012: parametergroup/ my-parameter-group</i>
Grup subnet	<i>arn:aws:memorydb: us-timur-1:123456789012: subnetgroup/ my-subnet-group</i>

MemoryDB menyediakan satu set operasi untuk bekerja dengan sumber daya MemoryDB. [Untuk daftar operasi yang tersedia, lihat Tindakan MemoryDB.](#)

Memahami kepemilikan sumber daya

Pemilik sumber daya adalah AWS akun yang membuat sumber daya. Artinya, pemilik sumber daya adalah AWS akun entitas utama yang mengotentikasi permintaan yang membuat sumber daya. Entitas utama dapat berupa akun root, pengguna IAM, atau peran IAM. Contoh berikut menggambarkan cara kerjanya:

- Misalkan Anda menggunakan kredensial akun root AWS akun Anda untuk membuat cluster. Dalam hal ini, AWS akun Anda adalah pemilik sumber daya. Di MemoryDB, sumber daya adalah cluster.
- Misalkan Anda membuat pengguna IAM di AWS akun Anda dan memberikan izin untuk membuat kluster ke pengguna tersebut. Dalam hal ini, pengguna dapat membuat cluster. Namun, AWS akun Anda, tempat pengguna berada, memiliki sumber daya cluster.
- Misalkan Anda membuat peran IAM di AWS akun Anda dengan izin untuk membuat kluster. Dalam hal ini, siapa pun yang dapat mengambil peran dapat membuat cluster. AWS Akun Anda, tempat peran tersebut berada, memiliki sumber daya cluster.

Mengelola akses ke sumber daya

Kebijakan izin menjelaskan siapa yang memiliki akses ke suatu objek. Bagian berikut menjelaskan opsi yang tersedia untuk membuat kebijakan izin.

Note

Bagian ini membahas penggunaan IAM dalam konteks MemoryDB. Bagian ini tidak memberikan informasi yang mendetail tentang layanan IAM. Untuk dokumentasi IAM lengkap, lihat [Apa yang Dimaksud dengan IAM?](#) dalam Panduan Pengguna IAM. Untuk informasi tentang sintaksis dan deskripsi kebijakan IAM, lihat [Referensi Kebijakan IAM AWS](#) dalam Panduan Pengguna IAM.

Kebijakan yang terlampir pada identitas IAM disebut sebagai kebijakan berbasis identitas (kebijakan IAM). Kebijakan yang dilampirkan pada sumber daya disebut sebagai kebijakan berbasis-sumber daya.

Topik

- [Kebijakan berbasis identitas \(kebijakan IAM\)](#)
- [Menentukan elemen kebijakan: Tindakan, efek, sumber daya, dan prinsipal](#)
- [Menentukan kondisi dalam kebijakan](#)

Kebijakan berbasis identitas (kebijakan IAM)

Anda dapat melampirkan kebijakan ke identitas IAM Anda. Misalnya, Anda dapat melakukan hal berikut:

- Melampirkan kebijakan izin pada pengguna atau grup dalam akun Anda – Akun administrator dapat menggunakan kebijakan izin yang terkait dengan pengguna tertentu untuk memberikan izin. Dalam hal ini, izin adalah untuk pengguna tersebut untuk membuat sumber daya MemoryDB, seperti cluster, grup parameter, atau grup keamanan.
- Melampirkan kebijakan izin pada peran (memberikan izin lintas akun) – Anda dapat melampirkan kebijakan izin berbasis identitas ke peran IAM untuk memberikan izin lintas akun. Misalnya, administrator di Akun A dapat membuat peran untuk memberikan izin lintas akun ke AWS akun lain (misalnya, Akun B) atau AWS layanan sebagai berikut:
 1. Administrator akun A membuat peran IAM dan melampirkan kebijakan izin ke peran ini yang memberikan izin pada sumber daya di akun A.
 2. Administrator akun A melampirkan kebijakan kepercayaan ke peran yang mengidentifikasi Akun B sebagai prinsipal yang dapat mengambil peran tersebut.

3. Administrator Akun B kemudian dapat mendelegasikan izin untuk mengambil peran kepada setiap pengguna di Akun B. Melakukan hal ini memungkinkan pengguna di Akun B untuk membuat atau mengakses sumber daya di Akun A. Dalam beberapa kasus, Anda mungkin ingin memberikan izin AWS layanan untuk mengambil peran tersebut. Untuk mendukung pendekatan ini, prinsipal dalam kebijakan kepercayaan juga dapat merupakan prinsipal layanan AWS .

Untuk informasi selengkapnya tentang penggunaan IAM untuk mendelegasikan izin, lihat [Manajemen Akses](#) dalam Panduan Pengguna IAM.

Berikut ini adalah contoh kebijakan yang memungkinkan pengguna untuk melakukan `DescribeClusters` tindakan untuk AWS akun Anda. MemoryDB juga mendukung identifikasi sumber daya tertentu menggunakan ARN sumber daya untuk tindakan API. (Pendekatan ini juga disebut sebagai izin tingkat sumber daya).

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "DescribeClusters",
    "Effect": "Allow",
    "Action": [
      "memorydb:DescribeClusters"],
    "Resource": resource-arn
  ]
}
```

Untuk informasi selengkapnya tentang penggunaan kebijakan berbasis identitas dengan MemoryDB, lihat [Menggunakan kebijakan berbasis identitas \(kebijakan IAM\) untuk MemoryDB](#) Untuk informasi selengkapnya tentang pengguna, grup, peran, dan izin, lihat [Identitas \(Pengguna, Grup, dan Peran\)](#) dalam Panduan Pengguna IAM.

Menentukan elemen kebijakan: Tindakan, efek, sumber daya, dan prinsipal

[Untuk setiap sumber daya MemoryDB \(lihat Sumber daya dan operasi MemoryDB\), layanan mendefinisikan satu set operasi API \(lihat Tindakan\).](#) Untuk memberikan izin untuk operasi API ini, MemoryDB mendefinisikan serangkaian tindakan yang dapat Anda tentukan dalam kebijakan. Misalnya, untuk sumber daya cluster MemoryDB, tindakan berikut didefinisikan: `CreateCluster`, `DeleteCluster`, dan `DescribeClusters` Operasi API dapat memerlukan izin untuk lebih dari satu tindakan.

Berikut adalah elemen-elemen kebijakan yang paling dasar:

- Sumber daya – Dalam kebijakan, Anda menggunakan Amazon Resource Name (ARN) untuk mengidentifikasi sumber daya yang diatur kebijakan. Untuk informasi selengkapnya, lihat [Sumber daya dan operasi MemoryDB](#).
- Tindakan – Anda menggunakan kata kunci tindakan untuk mengidentifikasi operasi sumber daya yang ingin Anda izinkan atau tolak. Misalnya, tergantung pada yang ditentukan `Effect`, `memorydb:CreateCluster` izin memungkinkan atau menolak izin pengguna untuk melakukan operasi `CreateCluster` MemoryDB.
- Efek – Anda menentukan efek ketika pengguna meminta tindakan tertentu—efek ini dapat berupa pemberian izin atau penolakan. Jika Anda tidak secara eksplisit memberikan akses ke (mengizinkan) sumber daya, akses akan ditolak secara implisit. Anda juga dapat secara eksplisit menolak akses ke sumber daya. Misalnya, Anda mungkin melakukannya untuk memastikan agar pengguna tidak dapat mengakses sumber daya, meskipun jika ada kebijakan berbeda yang memberikan akses.
- Prinsipal – Dalam kebijakan berbasis identitas (Kebijakan IAM), pengguna yang dilampiri kebijakan adalah prinsipal secara implisit. Untuk kebijakan berbasis sumber daya, Anda menentukan pengguna, akun, layanan, atau entitas lain yang diinginkan untuk menerima izin (berlaku hanya untuk kebijakan berbasis sumber daya).

Untuk mempelajari selengkapnya tentang sintaksis dan deskripsi kebijakan IAM, lihat [Referensi Kebijakan IAM AWS](#) dalam Panduan Pengguna IAM.

Untuk tabel yang menampilkan semua tindakan API MemoryDB, lihat. [Izin API MemoryDB: Referensi tindakan, sumber daya, dan kondisi](#)

Menentukan kondisi dalam kebijakan

Ketika Anda memberikan izin, Anda dapat menggunakan bahasa kebijakan IAM untuk menentukan kondisi ketika kebijakan harus berlaku. Misalnya, Anda mungkin ingin kebijakan diterapkan hanya setelah tanggal tertentu. Untuk informasi selengkapnya tentang menentukan kondisi dalam bahasa kebijakan, lihat [Kondisi](#) dalam Panduan Pengguna IAM.

Menggunakan kebijakan berbasis identitas (kebijakan IAM) untuk MemoryDB

Topik ini memberikan contoh kebijakan berbasis identitas di mana administrator akun dapat melampirkan kebijakan izin ke identitas IAM (yaitu, pengguna, grup, dan peran).

Important

Kami menyarankan Anda terlebih dahulu membaca topik yang menjelaskan konsep dasar dan opsi untuk mengelola akses ke sumber daya MemoryDB. Untuk informasi selengkapnya, lihat [Ikhtisar mengelola izin akses ke sumber daya MemoryDB Anda](#).

Bagian dalam topik ini membahas hal berikut:

- [Izin diperlukan untuk menggunakan konsol MemoryDB](#)
- [AWS kebijakan -managed \(telah ditentukan\) untuk MemoryDB](#)
- [Contoh kebijakan yang dikelola pelanggan](#)

Berikut adalah contoh kebijakan izin.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "AllowClusterPermissions",
    "Effect": "Allow",
    "Action": [
      "memorydb:CreateCluster",
      "memorydb:DescribeClusters",
      "memorydb:UpdateCluster"],
    "Resource": "*"
  },
  {
    "Sid": "AllowUserToPassRole",
    "Effect": "Allow",
    "Action": [ "iam:PassRole" ],
    "Resource": "arn:aws:iam::123456789012:role/EC2-roles-for-cluster"
  }
  ]
}
```

Kebijakan tersebut memiliki dua pernyataan:

- Pernyataan pertama memberikan izin untuk tindakan MemoryDB (`memorydb:CreateCluster`, `memorydb:DescribeClusters`, dan `memorydb:UpdateCluster`) pada kluster mana pun yang dimiliki oleh akun.
- Pernyataan kedua memberikan izin untuk tindakan IAM (`iam:PassRole`) pada nama peran IAM yang ditentukan pada akhir nilai `Resource`.

Kebijakan tidak menentukan elemen `Principal` karena dalam kebijakan berbasis identitas, Anda tidak menentukan prinsipal yang mendapatkan izin. Saat Anda melampirkan kebijakan kepada pengguna, pengguna tersebut menjadi prinsipal secara implisit. Saat Anda melampirkan kebijakan izin pada peran IAM, prinsipal yang diidentifikasi dalam kebijakan kepercayaan peran tersebut akan mendapatkan izin.

Untuk tabel yang menunjukkan semua tindakan API MemoryDB dan sumber daya yang diterapkan, lihat [Izin API MemoryDB: Referensi tindakan, sumber daya, dan kondisi](#)

Izin diperlukan untuk menggunakan konsol MemoryDB

Tabel referensi izin mencantumkan operasi API MemoryDB dan menunjukkan izin yang diperlukan untuk setiap operasi. Untuk informasi selengkapnya tentang operasi API MemoryDB, lihat [Izin API MemoryDB: Referensi tindakan, sumber daya, dan kondisi](#)

Untuk menggunakan konsol MemoryDB, pertama-tama berikan izin untuk tindakan tambahan seperti yang ditunjukkan dalam kebijakan izin berikut.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "MinPermsForMemDBConsole",
    "Effect": "Allow",
    "Action": [
      "memorydb:Describe*",
      "memorydb:List*",
      "ec2:DescribeAvailabilityZones",
      "ec2:DescribeVpcs",
      "ec2:DescribeAccountAttributes",
      "ec2:DescribeSecurityGroups",
      "cloudwatch:GetMetricStatistics",
      "cloudwatch:DescribeAlarms",
      "s3:ListAllMyBuckets",
    ]
  }]
}
```

```
        "sns:ListTopics",
        "sns:ListSubscriptions" ],
    "Resource": "*"
  }
]
```

Konsol MemoryDB membutuhkan izin tambahan ini karena alasan berikut:

- Izin untuk tindakan MemoryDB memungkinkan konsol untuk menampilkan sumber daya MemoryDB di akun.
- Konsol memerlukan izin untuk ec2 tindakan untuk menanyakan Amazon EC2 sehingga dapat menampilkan Availability Zone, VPC, grup keamanan, dan atribut akun.
- Izin untuk cloudwatch tindakan memungkinkan konsol mengambil CloudWatch metrik dan alarm Amazon, dan menampilkannya di konsol.
- Izin untuk sns tindakan memungkinkan konsol mengambil topik dan langganan Amazon Simple Notification Service (Amazon SNS), dan menampilkannya di konsol.

Contoh kebijakan yang dikelola pelanggan

Jika Anda tidak menggunakan kebijakan default dan memilih untuk menggunakan kebijakan yang dikelola kustom, pastikan salah satu dari dua hal berikut. Anda harus memiliki izin untuk memanggil `iam:createServiceLinkedRole` (untuk informasi selengkapnya, lihat [Contoh 4: Izinkan pengguna memanggil IAM API CreateServiceLinkedRole](#)). Atau Anda seharusnya telah membuat peran terkait layanan MemoryDB.

Bila dikombinasikan dengan izin minimum yang diperlukan untuk menggunakan konsol MemoryDB, contoh kebijakan di bagian ini memberikan izin tambahan. Contohnya juga relevan dengan AWS SDK dan AWS CLI Untuk informasi selengkapnya tentang izin apa yang diperlukan untuk menggunakan konsol MemoryDB, lihat. [Izin diperlukan untuk menggunakan konsol MemoryDB](#)

Untuk petunjuk pengaturan pengguna dan grup IAM, lihat [Membuat Pengguna dan Grup Administrator IAM Pertama Anda](#) dalam Panduan Pengguna IAM.

Important

Selalu uji kebijakan IAM Anda secara menyeluruh sebelum menggunakannya dalam produksi. Beberapa tindakan MemoryDB yang tampak sederhana dapat memerlukan tindakan lain untuk mendukungnya saat Anda menggunakan konsol MemoryDB. Misalnya,

`memorydb:CreateCluster` memberikan izin untuk membuat cluster MemoryDB. Namun, untuk melakukan operasi ini, konsol MemoryDB menggunakan sejumlah `Describe` dan `List` tindakan untuk mengisi daftar konsol.

Contoh

- [Contoh 1: Izinkan akses hanya-baca pengguna ke sumber daya MemoryDB](#)
- [Contoh 2: Izinkan pengguna untuk melakukan tugas administrator sistem MemoryDB umum](#)
- [Contoh 3: Izinkan pengguna mengakses semua tindakan API MemoryDB](#)
- [Contoh 4: Izinkan pengguna memanggil IAM API `CreateServiceLinkedRole`](#)

Contoh 1: Izinkan akses hanya-baca pengguna ke sumber daya MemoryDB

Kebijakan berikut memberikan izin untuk tindakan MemoryDB yang memungkinkan pengguna mencantumkan sumber daya. Biasanya, Anda melampirkan jenis kebijakan izin ini ke grup pengelola.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "MemDBUnrestricted",
    "Effect": "Allow",
    "Action": [
      "memorydb:Describe*",
      "memorydb:List*"
    ],
    "Resource": "*"
  }
]
```

Contoh 2: Izinkan pengguna untuk melakukan tugas administrator sistem MemoryDB umum

Tugas administrator sistem umum termasuk memodifikasi cluster, parameter, dan kelompok parameter. Administrator sistem mungkin juga ingin mendapatkan informasi tentang peristiwa MemoryDB. Kebijakan berikut memberikan izin pengguna untuk melakukan tindakan MemoryDB untuk tugas administrator sistem umum ini. Biasanya, Anda melampirkan jenis kebijakan izin ini ke grup administrator sistem.

```
{
```

```

"Version": "2012-10-17",
"Statement": [{
  "Sid": "MDBAllowSpecific",
  "Effect": "Allow",
  "Action": [
    "memorydb:UpdateCluster",
    "memorydb:DescribeClusters",
    "memorydb:DescribeEvents",
    "memorydb:UpdateParameterGroup",
    "memorydb:DescribeParameterGroups",
    "memorydb:DescribeParameters",
    "memorydb:ResetParameterGroup", ],
  "Resource": "*"
}
]
}

```

Contoh 3: Izinkan pengguna mengakses semua tindakan API MemoryDB

Kebijakan berikut memungkinkan pengguna untuk mengakses semua tindakan MemoryDB. Sebaiknya Anda memberikan jenis kebijakan izin ini hanya untuk pengguna administrator.

```

{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "MDBAllowAll",
    "Effect": "Allow",
    "Action": [
      "memorydb:*" ],
    "Resource": "*"
  }
]
}

```

Contoh 4: Izinkan pengguna memanggil IAM API CreateServiceLinkedRole

Kebijakan berikut mengizinkan pengguna untuk memanggil API CreateServiceLinkedRole IAM. Kami menyarankan Anda memberikan jenis kebijakan izin ini kepada pengguna yang memanggil operasi MemoryDB mutatif.

```

{
  "Version": "2012-10-17",

```



```
"Statement":[
  {
    "Sid":"CreateSLRAllows",
    "Effect":"Allow",
    "Action":[
      "iam:CreateServiceLinkedRole"
    ],
    "Resource":"*",
    "Condition":{"
      "StringLike":{"
        "iam:AWS ServiceName":"memorydb.amazonaws.com"
      }
    }
  }
]
```

Izin tingkat sumber daya

Anda dapat membatasi cakupan izin dengan menentukan sumber daya dalam kebijakan IAM. Banyak tindakan AWS CLI API mendukung jenis sumber daya yang bervariasi tergantung pada perilaku tindakan. Setiap pernyataan kebijakan IAM memberikan izin untuk tindakan yang dilakukan pada sumber daya. Saat tindakan tersebut tidak dilakukan pada sumber daya yang disebutkan, atau saat Anda memberikan izin untuk melakukan tindakan pada semua sumber daya, maka nilai sumber daya dalam kebijakan tersebut adalah wildcard (*). Untuk banyak tindakan API, Anda dapat membatasi sumber daya yang dapat diubah oleh pengguna dengan menentukan Amazon Resource Name (ARN) sumber daya tersebut, atau pola ARN yang cocok dengan beberapa sumber daya. Untuk membatasi izin berdasarkan sumber daya, tentukan sumber daya berdasarkan ARN.

Format ARN Sumber Daya MemoryDB

Note

Agar izin di tingkat sumber daya menjadi efektif, nama sumber daya pada string ARN harus huruf kecil.

- *Pengguna* - *arn:aws:memorydb: us-timur- 1:123456789012: pengguna/pengguna1*
- *ACL* - *arn:aws:memorydb: us-timur- 1:123456789012: acl/my-acl*
- *Cluster* - *arn:aws:memorydb: us-timur- 1:123456789012: cluster/my-cluster*

- *Snapshot* – `arn:aws:memorydb:us-timur-1:123456789012:snapshot/snapshot saya`
- *Kelompok parameter* - `arn:aws:memorydb:us-timur-1:123456789012:parametergroup/my-parameter-group`
- *Grup subnet* - `arn:aws:memorydb:us-timur-1:123456789012:subnetgroup/my-subnet-group`

Contoh

- [Contoh 1: Izinkan pengguna akses penuh ke jenis sumber daya MemoryDB tertentu](#)
- [Contoh 2: Tolak akses pengguna ke cluster.](#)

Contoh 1: Izinkan pengguna akses penuh ke jenis sumber daya MemoryDB tertentu

Kebijakan berikut secara eksplisit memungkinkan akses `account-id` penuh yang ditentukan ke semua sumber daya grup subnet tipe, grup keamanan, dan klaster.

```
{
  "Sid": "Example1",
  "Effect": "Allow",
  "Action": "memorydb:*",
  "Resource": [
    "arn:aws:memorydb:us-east-1:account-id:subnetgroup/*",
    "arn:aws:memorydb:us-east-1:account-id:securitygroup/*",
    "arn:aws:memorydb:us-east-1:account-id:cluster/*"
  ]
}
```

Contoh 2: Tolak akses pengguna ke cluster.

Contoh berikut secara eksplisit menyangkal `account-id` akses yang ditentukan ke cluster tertentu.

```
{
  "Sid": "Example2",
  "Effect": "Deny",
  "Action": "memorydb:*",
  "Resource": [
    "arn:aws:memorydb:us-east-1:account-id:cluster/name"
  ]
}
```

Menggunakan Peran Tertaut Layanan untuk MemoryDB

[MemoryDB menggunakan peran AWS Identity and Access Management terkait layanan \(IAM\).](#)

Peran terkait layanan adalah jenis unik peran IAM yang ditautkan langsung ke AWS layanan, seperti MemoryDB. Peran terkait layanan MemoryDB telah ditentukan sebelumnya oleh MemoryDB. Peran tersebut menyertakan semua izin yang diperlukan layanan untuk memanggil layanan AWS atas nama klaster Anda.

Peran terkait layanan membuat pengaturan MemoryDB lebih mudah karena Anda tidak perlu menambahkan izin yang diperlukan secara manual. Peran sudah ada dalam AWS akun Anda tetapi ditautkan ke kasus penggunaan MemoryDB dan memiliki izin yang telah ditentukan sebelumnya. Hanya MemoryDB yang dapat mengambil peran ini, dan hanya peran ini yang dapat menggunakan kebijakan izin yang telah ditentukan sebelumnya. Anda dapat menghapus peran tersebut hanya setelah pertama kali menghapus sumber dayanya yang terkait. Ini melindungi sumber daya MemoryDB Anda karena Anda tidak dapat secara tidak sengaja menghapus izin yang diperlukan untuk mengakses sumber daya.

Untuk informasi tentang layanan lain yang mendukung peran terkait layanan, lihat [Layanan AWS yang bisa digunakan dengan IAM](#) dan carilah layanan yang memiliki opsi Ya di kolom Peran Terkait Layanan. Pilih Ya bersama tautan untuk melihat dokumentasi peran tertaut layanan untuk layanan tersebut.

Daftar Isi

- [Izin Peran Tertaut Layanan untuk MemoryDB](#)
- [Membuat Peran Terkait Layanan \(IAM\)](#)
 - [Membuat Peran Terkait Layanan \(Konsol IAM\)](#)
 - [Membuat Peran Terkait Layanan \(CLI IAM\)](#)
 - [Membuat Peran Terkait Layanan \(API IAM\)](#)
- [Mengedit Deskripsi Peran Tertaut Layanan untuk MemoryDB](#)
 - [Mengedit Deskripsi Peran Terkait Layanan \(Konsol IAM\)](#)
 - [Mengedit Deskripsi Peran Terkait Layanan \(CLI IAM\)](#)
 - [Mengedit Deskripsi Peran Terkait Layanan \(API IAM\)](#)
- [Menghapus Peran Tertaut Layanan untuk MemoryDB](#)
 - [Membersihkan Peran Terkait Layanan](#)
 - [Menghapus Peran Terkait Layanan \(Konsol IAM\)](#)

- [Menghapus Peran Terkait Layanan \(CLI IAM\)](#)
- [Menghapus Peran Terkait Layanan \(API IAM\)](#)

Izin Peran Tertaut Layanan untuk MemoryDB

MemoryDB menggunakan peran terkait layanan bernama `AWSServiceRoleForMemoryDB`. Kebijakan ini memungkinkan MemoryDB mengelola AWS sumber daya atas nama Anda sebagaimana diperlukan untuk mengelola cluster Anda.

Kebijakan izin peran `AWSServiceRoleForMemoryDB` terkait layanan memungkinkan MemoryDB menyelesaikan tindakan berikut pada sumber daya yang ditentukan:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:CreateTags"
      ],
      "Resource": "arn:aws:ec2:*:*:network-interface/*",
      "Condition": {
        "StringEquals": {
          "ec2:CreateAction": "CreateNetworkInterface"
        },
        "ForAllValues:StringEquals": {
          "aws:TagKeys": [
            "AmazonMemoryDBManaged"
          ]
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterface"
      ],
      "Resource": [
        "arn:aws:ec2:*:*:network-interface/*",
        "arn:aws:ec2:*:*:subnet/*",
        "arn:aws:ec2:*:*:security-group*"
      ]
    }
  ]
}
```

```
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DeleteNetworkInterface",
        "ec2:ModifyNetworkInterfaceAttribute"
      ],
      "Resource": "arn:aws:ec2:*:*:network-interface/*",
      "Condition": {
        "StringEquals": {
          "ec2:ResourceTag/AmazonMemoryDBManaged": "true"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DeleteNetworkInterface",
        "ec2:ModifyNetworkInterfaceAttribute"
      ],
      "Resource": "arn:aws:ec2:*:*:security-group/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcs"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "cloudwatch:PutMetricData"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "cloudwatch:namespace": "AWS/MemoryDB"
        }
      }
    }
  ]
}
```

```

    }
  ]
}

```

Untuk informasi selengkapnya, lihat [AWS kebijakan terkelola: MemoryDB ServiceRolePolicy](#).

Untuk mengizinkan entitas IAM membuat peran terkait AWSServiceRoleForMemoryDB layanan

Tambahkan pernyataan kebijakan berikut ini ke izin untuk entitas IAM:

```

{
  "Effect": "Allow",
  "Action": [
    "iam:CreateServiceLinkedRole",
    "iam:PutRolePolicy"
  ],
  "Resource": "arn:aws:iam::*:role/aws-service-role/memorydb.amazonaws.com/AWSServiceRoleForMemoryDB*",
  "Condition": {"StringLike": {"iam:AWS ServiceName": "memorydb.amazonaws.com"}}
}

```

Untuk mengizinkan entitas IAM menghapus peran terkait AWSServiceRoleForMemoryDB layanan

Tambahkan pernyataan kebijakan berikut ini ke izin untuk entitas IAM:

```

{
  "Effect": "Allow",
  "Action": [
    "iam>DeleteServiceLinkedRole",
    "iam:GetServiceLinkedRoleDeletionStatus"
  ],
  "Resource": "arn:aws:iam::*:role/aws-service-role/memorydb.amazonaws.com/AWSServiceRoleForMemoryDB*",
  "Condition": {"StringLike": {"iam:AWS ServiceName": "memorydb.amazonaws.com"}}
}

```

Atau, Anda dapat menggunakan kebijakan AWS terkelola untuk menyediakan akses penuh ke MemoryDB.

Membuat Peran Terkait Layanan (IAM)

Anda dapat membuat peran terkait layanan menggunakan konsol IAM, CLI, atau API.

Membuat Peran Terkait Layanan (Konsol IAM)

Anda dapat menggunakan konsol IAM untuk membuat peran terkait layanan.

Untuk membuat peran terkait layanan (konsol)

1. Masuk ke AWS Management Console dan buka konsol IAM di <https://console.aws.amazon.com/iam/>.
2. Di panel navigasi kiri konsol IAM, pilih Peran. Kemudian pilih Buat peran baru.
3. Di bagian Pilih jenis entitas tepercaya, pilih Layanan AWS .
4. Di bawah Atau pilih layanan untuk melihat kasus penggunaannya, pilih MemoryDB.
5. Pilih Berikutnya: Izin.
6. Di bagian Nama kebijakan, perhatikan bahwa MemoryDBServiceRolePolicy diperlukan untuk peran ini. Pilih Selanjutnya: Tag.
7. Perhatikan bahwa tag tidak didukung untuk peran Tertaut-Layanan. Pilih Selanjutnya: Tinjau.
8. (Opsional) Untuk Deskripsi peran, edit deskripsi untuk peran terkait layanan baru.
9. Tinjau peran, lalu pilih Buat peran.

Membuat Peran Terkait Layanan (CLI IAM)

Anda dapat menggunakan operasi IAM dari AWS Command Line Interface untuk membuat peran terkait layanan. Peran ini dapat mencakup kebijakan kepercayaan dan kebijakan terkait yang diperlukan oleh layanan untuk mengambil peran tersebut.

Untuk membuat peran terkait layanan (CLI)

Gunakan operasi berikut:

```
$ aws iam create-service-linked-role --aws-service-name memorydb.amazonaws.com
```

Membuat Peran Terkait Layanan (API IAM)

Anda dapat menggunakan API IAM untuk membuat peran terkait layanan. Peran ini dapat berisi kebijakan kepercayaan dan kebijakan terkait yang diperlukan oleh layanan untuk mengambil peran tersebut.

Untuk membuat peran terkait layanan (API)

Gunakan panggilan API [CreateServiceLinkedRole](#). Dalam permintaan, sebutkan nama layanan `memorydb.amazonaws.com`.

Mengedit Deskripsi Peran Tertaut Layanan untuk MemoryDB

MemoryDB tidak memungkinkan Anda untuk mengedit peran terkait `AWSServiceRoleForMemoryDB` layanan. Setelah Anda membuat peran terkait layanan, Anda tidak dapat mengubah nama peran karena berbagai entitas mungkin mereferensikan peran tersebut. Namun, Anda dapat menyunting Deskripsi peran menggunakan IAM.

Mengedit Deskripsi Peran Terkait Layanan (Konsol IAM)

Anda dapat menggunakan konsol IAM untuk mengedit deskripsi peran terkait layanan.

Untuk mengedit deskripsi peran terkait layanan (konsol)

1. Di panel navigasi kiri konsol IAM, pilih Peran.
2. Pilih nama peran yang akan diubah.
3. Di ujung kanan Deskripsi peran, pilih Edit.
4. Masukkan deskripsi baru di kotak, lalu pilih Simpan.

Mengedit Deskripsi Peran Terkait Layanan (CLI IAM)

Anda dapat menggunakan operasi IAM dari AWS Command Line Interface untuk mengedit deskripsi peran terkait layanan.

Untuk mengubah deskripsi peran terkait layanan (CLI)

1. (Opsional) Untuk melihat deskripsi saat ini untuk peran, gunakan AWS CLI untuk operasi [get-role](#) IAM.

Example

```
$ aws iam get-role --role-name AWSServiceRoleForMemoryDB
```

Gunakan nama peran, bukan ARN, untuk mengacu ke peran itu dengan operasi CLI. Misalnya, jika peran memiliki ARN berikut: `arn:aws:iam::123456789012:role/myrole`, peran akan dirujuk sebagai **myrole**.

2. Untuk memperbarui deskripsi peran terkait layanan, gunakan operasi AWS CLI untuk IAM. [update-role-description](#)

Untuk Linux, macOS, atau Unix:

```
$ aws iam update-role-description \  
  --role-name AWSServiceRoleForMemoryDB \  
  --description "new description"
```

Untuk Windows:

```
$ aws iam update-role-description ^  
  --role-name AWSServiceRoleForMemoryDB ^  
  --description "new description"
```

Mengedit Deskripsi Peran Terkait Layanan (API IAM)

Anda dapat menggunakan API IAM untuk mengedit deskripsi peran terkait layanan.

Untuk mengubah deskripsi peran terkait layanan (API)

1. (Opsional) Untuk melihat deskripsi peran saat ini, gunakan operasi API IAM [GetRole](#).

Example

```
https://iam.amazonaws.com/  
  ?Action=GetRole  
  &RoleName=AWSServiceRoleForMemoryDB  
  &Version=2010-05-08  
  &AUTHPARAMS
```

2. Untuk memperbarui deskripsi peran, gunakan operasi API IAM [UpdateRoleDescription](#).

Example

```
https://iam.amazonaws.com/  
  ?Action=UpdateRoleDescription  
  &RoleName=AWSServiceRoleForMemoryDB  
  &Version=2010-05-08  
  &Description="New description"
```

Menghapus Peran Tertaut Layanan untuk MemoryDB

Jika Anda tidak perlu lagi menggunakan fitur atau layanan yang memerlukan peran terkait layanan, kami merekomendasikan Anda menghapus peran tersebut. Dengan begitu, Anda tidak perlu lagi memantau atau memelihara entitas yang tidak digunakan. Namun, Anda harus membersihkan peran terkait layanan sebelum dapat menghapusnya.

MemoryDB tidak menghapus peran terkait layanan untuk Anda.

Membersihkan Peran Terkait Layanan

Sebelum Anda dapat menggunakan IAM untuk menghapus peran terkait layanan, konfirmasi terlebih dahulu bahwa peran tersebut tidak memiliki sumber daya (kluster) yang terkait dengannya.

Untuk memastikan peran terkait layanan memiliki sesi aktif di konsol IAM

1. Masuk ke AWS Management Console dan buka konsol IAM di <https://console.aws.amazon.com/iam/>.
2. Di panel navigasi kiri konsol IAM, pilih Peran. Kemudian pilih nama (bukan kotak centang) AWSServiceRoleForMemoryDB peran.
3. Di halaman Ringkasan untuk peran yang dipilih, pilih tab Penasihat Akses.
4. Di tab Penasihat Akses, tinjau aktivitas terbaru untuk peran terkait layanan tersebut.

Untuk menghapus sumber daya MemoryDB yang membutuhkan AWSServiceRoleForMemoryDB (konsol)

- Untuk menghapus kluster, lihat referensi berikut:
 - [Menggunakan AWS Management Console](#)
 - [Menggunakan AWS CLI](#)
 - [Menggunakan MemoryDB API](#)

Menghapus Peran Terkait Layanan (Konsol IAM)

Anda dapat menggunakan konsol IAM untuk menghapus sebuah peran terkait layanan.

Untuk menghapus peran terkait layanan (konsol)

1. Masuk ke AWS Management Console dan buka konsol IAM di <https://console.aws.amazon.com/iam/>.

2. Di panel navigasi kiri konsol IAM, pilih Peran. Kemudian, pilih kotak centang di sebelah nama peran yang ingin dihapus, bukan nama atau baris itu sendiri.
3. Untuk Tindakan peran di bagian atas halaman, pilih Hapus peran.
4. Di halaman konfirmasi, tinjau data layanan yang terakhir diakses, yang menunjukkan kapan masing-masing peran yang dipilih terakhir mengakses AWS layanan. Hal ini membantu Anda mengonfirmasi aktif tidaknya peran tersebut saat ini. Jika Anda ingin melanjutkan, pilih Ya, Hapus guna mengirimkan peran terkait layanan untuk penghapusan.
5. Perhatikan notifikasi konsol IAM untuk memantau progres penghapusan peran terkait layanan. Karena penghapusan peran terkait layanan IAM bersifat asinkron, setelah Anda mengirimkan peran tersebut untuk penghapusan, tugas penghapusan dapat berhasil atau gagal. Jika tugas tersebut gagal, Anda dapat memilih Lihat detail atau Lihat Sumber Daya dari notifikasi untuk mempelajari alasan gagalnya penghapusan.

Menghapus Peran Terkait Layanan (CLI IAM)

Anda dapat menggunakan operasi IAM dari AWS Command Line Interface untuk menghapus peran terkait layanan.

Untuk menghapus peran terkait layanan (CLI)

1. Jika Anda tidak tahu nama peran terkait layanan yang ingin dihapus, masukkan perintah berikut. Perintah ini menampilkan daftar peran dan Amazon Resource Name (ARN) di akun Anda.

```
$ aws iam get-role --role-name role-name
```

Gunakan nama peran, bukan ARN, untuk merujuk ke peran dengan operasi CLI. Misalnya, jika peran memiliki ARN `arn:aws:iam::123456789012:role/myrole`, peran akan dirujuk sebagai **myrole**.

2. Karena peran terkait layanan tidak dapat dihapus jika sedang digunakan atau memiliki sumber daya terkait, Anda harus mengirimkan permintaan penghapusan. Permintaan tersebut dapat ditolak jika kondisi ini tidak terpenuhi. Anda harus menangkap `deletion-task-id` dari tanggapan untuk memeriksa status tugas penghapusan. Masukkan perintah berikut untuk mengirimkan permintaan penghapusan peran terkait layanan.

```
$ aws iam delete-service-linked-role --role-name role-name
```

3. Masukkan perintah berikut untuk memeriksa status tugas penghapusan.

```
$ aws iam get-service-linked-role-deletion-status --deletion-task-id deletion-task-id
```

Status tugas penghapusan dapat berupa NOT_STARTED, IN_PROGRESS, SUCCEEDED, atau FAILED. Jika penghapusan gagal, panggilan akan mengembalikan alasan kegagalan panggilan agar Anda dapat memecahkan masalah.

Menghapus Peran Terkait Layanan (API IAM)

Anda dapat menggunakan API IAM untuk menghapus peran terkait layanan.

Untuk menghapus peran terkait layanan (API)

1. Untuk mengirimkan permintaan penghapusan untuk peran terkait layanan, panggil [DeleteServiceLinkedRole](#). Di permintaan tersebut, tentukan nama peran.

Karena peran terkait layanan tidak dapat dihapus jika sedang digunakan atau memiliki sumber daya terkait, Anda harus mengirimkan permintaan penghapusan. Permintaan tersebut dapat ditolak jika kondisi ini tidak terpenuhi. Anda harus menangkap DeletionTaskId dari tanggapan untuk memeriksa status tugas penghapusan.

2. Untuk memeriksa status penghapusan, panggil [GetServiceLinkedRoleDeletionStatus](#). Di permintaan tersebut, tentukan DeletionTaskId.

Status tugas penghapusan dapat berupa NOT_STARTED, IN_PROGRESS, SUCCEEDED, atau FAILED. Jika penghapusan gagal, panggilan akan mengembalikan alasan kegagalan panggilan agar Anda dapat memecahkan masalah.

AWS kebijakan terkelola untuk MemoryDB

Untuk menambahkan izin ke pengguna, grup, dan peran, lebih mudah menggunakan kebijakan AWS terkelola daripada menulis kebijakan sendiri. Dibutuhkan waktu dan keahlian untuk [membuat kebijakan yang dikelola pelanggan IAM](#) yang hanya memberi tim Anda izin yang mereka butuhkan. Untuk memulai dengan cepat, Anda dapat menggunakan kebijakan AWS terkelola kami. Kebijakan ini mencakup kasus penggunaan umum dan tersedia di AWS akun Anda. Untuk informasi selengkapnya tentang kebijakan AWS [AWS terkelola, lihat kebijakan terkelola](#) di Panduan Pengguna IAM.

AWS layanan memelihara dan memperbarui kebijakan AWS terkelola. Anda tidak dapat mengubah izin dalam kebijakan AWS terkelola. Layanan terkadang menambahkan izin tambahan ke kebijakan yang dikelola AWS untuk mendukung fitur-fitur baru. Jenis pembaruan ini akan memengaruhi semua identitas (pengguna, grup, dan peran) di mana kebijakan tersebut dilampirkan. Layanan kemungkinan besar akan memperbarui kebijakan yang dikelola AWS saat ada fitur baru yang diluncurkan atau saat ada operasi baru yang tersedia. Layanan tidak menghapus izin dari kebijakan AWS terkelola, sehingga pembaruan kebijakan tidak akan merusak izin yang ada.

Selain itu, AWS mendukung kebijakan terkelola untuk fungsi pekerjaan yang mencakup beberapa layanan. Misalnya, kebijakan `ReadOnlyAccess` AWS terkelola menyediakan akses hanya-baca ke semua AWS layanan dan sumber daya. Saat layanan meluncurkan fitur baru, AWS menambahkan izin hanya-baca untuk operasi dan sumber daya baru. Untuk melihat daftar dan deskripsi dari kebijakan fungsi tugas, lihat [kebijakan yang dikelola AWS untuk fungsi tugas](#) di Panduan Pengguna IAM.

AWS kebijakan terkelola: MemoryDB ServiceRolePolicy

Anda tidak dapat melampirkan kebijakan `ServiceRolePolicy` AWS terkelola MemoryDB ke identitas di akun Anda. Kebijakan ini merupakan bagian dari peran terkait layanan AWS MemoryDB. Peran ini memungkinkan layanan untuk mengelola antarmuka jaringan dan grup keamanan di akun Anda.

MemoryDB menggunakan izin dalam kebijakan ini untuk mengelola grup keamanan EC2 dan antarmuka jaringan. Ini diperlukan untuk mengelola cluster MemoryDB.

Detail izin

Kebijakan ini mencakup izin berikut.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:CreateTags"
      ],
      "Resource": "arn:aws:ec2:*:*:network-interface/*",
    }
  ]
}
```

```

    "Condition": {
      "StringEquals": {
        "ec2:CreateAction": "CreateNetworkInterface"
      },
      "ForAllValues:StringEquals": {
        "aws:TagKeys": [
          "AmazonMemoryDBManaged"
        ]
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "ec2:CreateNetworkInterface"
    ],
    "Resource": [
      "arn:aws:ec2:*:*:network-interface/*",
      "arn:aws:ec2:*:*:subnet/*",
      "arn:aws:ec2:*:*:security-group*"
    ]
  },
  {
    "Effect": "Allow",
    "Action": [
      "ec2>DeleteNetworkInterface",
      "ec2:ModifyNetworkInterfaceAttribute"
    ],
    "Resource": "arn:aws:ec2:*:*:network-interface/*",
    "Condition": {
      "StringEquals": {
        "ec2:ResourceTag/AmazonMemoryDBManaged": "true"
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "ec2>DeleteNetworkInterface",
      "ec2:ModifyNetworkInterfaceAttribute"
    ],
    "Resource": "arn:aws:ec2:*:*:security-group*"
  },
  {

```

```

    "Effect": "Allow",
    "Action": [
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcs"
    ],
    "Resource": "*"
},
{
    "Effect": "Allow",
    "Action": [
        "cloudwatch:PutMetricData"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "cloudwatch:namespace": "AWS/MemoryDB"
        }
    }
}
]
}

```

AWS kebijakan -managed (telah ditentukan) untuk MemoryDB

AWS mengatasi banyak kasus penggunaan umum dengan menyediakan kebijakan IAM mandiri yang dibuat dan dikelola oleh AWS. Kebijakan terkelola memberikan izin yang diperlukan untuk kasus penggunaan umum sehingga Anda tidak perlu menyelidiki izin apa yang diperlukan. Untuk informasi selengkapnya, lihat [Kebijakan Terkelola AWS](#) dalam Panduan Pengguna IAM.

Kebijakan AWS terkelola berikut, yang dapat Anda lampirkan ke pengguna di akun Anda, khusus untuk MemoryDB:

AmazonMemoryDB ReadOnlyAccess

Anda dapat melampirkan kebijakan `AmazonMemoryDBReadOnlyAccess` ke identitas IAM Anda. Kebijakan ini memberikan izin administratif yang memungkinkan akses hanya-baca ke semua sumber daya MemoryDB.

`AmazonMemoryDB ReadOnlyAccess` - Memberikan akses hanya-baca ke sumber daya MemoryDB.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "memorydb:Describe*",
      "memorydb:List*"
    ],
    "Resource": "*"
  }]
}
```

AmazonMemoryDB FullAccess

Anda dapat melampirkan kebijakan AmazonMemoryDBFullAccess ke identitas IAM Anda. Kebijakan ini memberikan izin administratif yang memungkinkan akses penuh ke semua sumber daya MemoryDB.

AmazonMemoryDB FullAccess - Memberikan akses penuh ke sumber daya MemoryDB.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "memorydb:*",
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": "iam:CreateServiceLinkedRole",
    "Resource": "arn:aws:iam::*:role/aws-service-role/memorydb.amazonaws.com/
AWSServiceRoleForMemoryDB",
    "Condition": {
      "StringLike": {
        "iam:AWSServiceName": "memorydb.amazonaws.com"
      }
    }
  }
  ]
}
```


Anda juga dapat membuat kebijakan IAM kustom Anda sendiri untuk mengizinkan izin untuk tindakan API MemoryDB. Anda dapat melampirkan kebijakan-kebijakan kustom ini ke pengguna IAM atau grup yang memerlukan izin-izin tersebut.

Pembaruan MemoryDB ke AWS kebijakan terkelola

Lihat detail tentang pembaruan kebijakan AWS terkelola untuk MemoryDB sejak layanan ini mulai melacak perubahan ini. Untuk peringatan otomatis tentang perubahan pada halaman ini, berlangganan umpan RSS di halaman riwayat Dokumen MemoryDB.

Perubahan	Deskripsi	Tanggal
AmazonMemoryDB FullAccess — Menambahkan kebijakan	MemoryDB menambahkan izin baru untuk mendeskripsikan dan mencantumkan sumber daya yang didukung. Izin ini diperlukan untuk MemoryDB untuk menanyakan semua sumber daya yang didukung dalam akun.	10/07/2021
AmazonMemoryDB ReadOnlyAccess — Menambahkan kebijakan	MemoryDB menambahkan izin baru untuk mendeskripsikan dan mencantumkan sumber daya yang didukung. Izin ini diperlukan untuk MemoryDB untuk membuat aplikasi berbasis akun dengan menanyakan semua sumber daya yang didukung dalam akun.	10/07/2021
MemoryDB mulai melacak perubahan	Peluncuran layanan	8/19/2021

Izin API MemoryDB: Referensi tindakan, sumber daya, dan kondisi

Saat Anda mengatur [kontrol akses](#) dan menulis kebijakan izin untuk dilampirkan ke kebijakan IAM (baik berbasis identitas atau berbasis sumber daya), gunakan tabel berikut sebagai referensi. Tabel mencantumkan setiap operasi API MemoryDB dan tindakan terkait yang dapat Anda berikan izin untuk melakukan tindakan. Anda menentukan tindakan dalam bidang `Action` kebijakan, dan Anda menentukan nilai sumber daya pada bidang `Resource` kebijakan. Kecuali dinyatakan sebaliknya, sumber daya wajib ditentukan. Beberapa bidang mencakup sumber daya wajib dan sumber daya opsional. Jika tidak terdapat ARN sumber daya, sumber daya dalam kebijakan adalah wildcard (*).

Note

Untuk menentukan tindakan, gunakan awalan `memorydb:` diikuti dengan nama operasi API (misalnya, `memorydb:DescribeClusters`).

Pencatatan dan pemantauan

Pemantauan adalah bagian penting dari menjaga keandalan, ketersediaan, dan kinerja MemoryDB dan solusi Anda yang lain AWS . AWS menyediakan alat pemantauan berikut untuk menonton MemoryDB, melaporkan ketika ada sesuatu yang salah, dan mengambil tindakan otomatis bila perlu:

- Amazon CloudWatch memantau AWS sumber daya Anda dan aplikasi yang Anda jalankan AWS secara real time. Anda dapat mengumpulkan dan melacak metrik, membuat dasbor yang disesuaikan, dan mengatur alarm yang memberi tahu Anda atau mengambil tindakan saat metrik tertentu mencapai ambang batas yang ditentukan. Misalnya, Anda dapat CloudWatch melacak CPU penggunaan atau metrik lain dari EC2 instans Amazon Anda dan secara otomatis meluncurkan instans baru bila diperlukan. Untuk informasi selengkapnya, lihat [Panduan CloudWatch Pengguna Amazon](#).
- Amazon CloudWatch Logs memungkinkan Anda memantau, menyimpan, dan mengakses file log Anda dari EC2 instans Amazon CloudTrail, dan sumber lainnya. CloudWatch Log dapat memantau informasi dalam file log dan memberi tahu Anda ketika ambang batas tertentu terpenuhi. Anda juga dapat mengarsipkan data log dalam penyimpanan yang sangat durabel. Untuk informasi selengkapnya, lihat [Panduan Pengguna Amazon CloudWatch Logs](#).
- AWS CloudTrail menangkap API panggilan dan peristiwa terkait yang dibuat oleh atau atas nama AWS akun Anda dan mengirimkan file log ke bucket Amazon S3 yang Anda tentukan. Anda dapat mengidentifikasi pengguna dan akun mana yang dipanggil AWS, alamat IP sumber dari mana

panggilan dilakukan, dan kapan panggilan terjadi. Untuk informasi selengkapnya, silakan lihat [Panduan Pengguna AWS CloudTrail](#).

Memonitor MemoryDB dengan Amazon CloudWatch

Anda dapat memantau penggunaan MemoryDB CloudWatch, yang mengumpulkan data mentah dan memrosesnya menjadi metrik yang dapat dibaca, mendekati waktu nyata. Statistik ini disimpan untuk jangka waktu 15 bulan, sehingga Anda dapat mengakses informasi historis dan mendapatkan perspektif yang lebih baik tentang performa aplikasi atau layanan web Anda. Anda juga dapat mengatur alarm yang memperhatikan ambang batas tertentu dan mengirim notifikasi atau mengambil tindakan saat ambang batas tersebut terpenuhi. Untuk informasi selengkapnya, lihat [Panduan CloudWatch Pengguna Amazon](#).

Bagian berikut mencantumkan metrik dan dimensi untuk MemoryDB.

Topik

- [Metrik Tingkat Host](#)
- [Metrik untuk MemoryDB](#)
- [Metrik Apa yang Harus Saya Pantau?](#)
- [Memilih Statistik dan Periode Metrik](#)
- [CloudWatch Metrik pemantauan](#)

Metrik Tingkat Host

AWS/MemoryDBNamespace menyertakan metrik tingkat host berikut untuk masing-masing node.

Lihat Juga

- [Metrik untuk MemoryDB](#)

Metrik	Deskripsi	Unit
CPUUtilization	Persentase CPU pemanfaatan untuk seluruh host. Karena Redis OSS adalah single-threaded, dan kami sarankan Anda memantau	Persen

Metrik	Deskripsi	Unit
	EngineCPUUtilization metrik untuk node dengan 4 atau lebih. vCPUs	
FreeableMemory	Jumlah memori kosong yang tersedia di host. Ini berasal dari RAM, buffer, dan bahwa OS melaporkan sebagai freeable.	Byte
NetworkBytesIn	Jumlah byte yang telah dibaca oleh host dari jaringan.	Byte
NetworkBytesOut	Jumlah byte yang dikirimkan ke semua antarmuka jaringan oleh instans.	Byte
NetworkPacketsIn	Jumlah paket yang diterima di semua antarmuka jaringan oleh instans. Metrik ini mengidentifikasi volume lalu lintas masuk dari segi jumlah paket pada satu instans tunggal.	Hitungan
NetworkPacketsOut	Jumlah paket yang dikirimkan di semua antarmuka jaringan oleh instans. Metrik ini mengidentifikasi volume lalu lintas keluar dari segi jumlah paket pada satu instans tunggal.	Hitung
NetworkBandwidthIn AllowanceExceeded	Jumlah paket yang di-shaping karena bandwidth agregat masuk melebihi nilai maksimum untuk instans.	Hitung
NetworkConntrackAllowanceExceeded	Jumlah paket yang di-shaping karena pelacakan koneksi melebihi nilai maksimum untuk instans dan koneksi baru tidak dapat dibuat. Hal ini dapat mengakibatkan hilangnya paket untuk lalu lintas ke atau dari instans.	Hitung
NetworkBandwidthOut AllowanceExceeded	Jumlah paket yang di-shaping karena bandwidth agregat keluar melebihi nilai maksimum untuk instans.	Hitung

Metrik	Deskripsi	Unit
NetworkPacketsPerSecondAllowanceExceeded	Jumlah paket yang di-shaping karena paket dua arah per detik melebihi nilai maksimum untuk instans.	Hitung
NetworkMaxBytesIn	Semburan maksimum per detik byte yang diterima dalam setiap menit.	Byte
NetworkMaxBytesOut	Semburan maksimum per detik byte yang ditransmisikan dalam setiap menit.	Byte
NetworkMaxPacketsIn	Semburan maksimum per detik dari paket yang diterima dalam setiap menit.	Hitung
NetworkMaxPacketsOut	Semburan maksimum per detik paket yang ditransmisikan dalam setiap menit.	Hitung
SwapUsage	Jumlah swap yang digunakan pada host.	Byte

Metrik untuk MemoryDB

AWS/MemoryDBNamespace mencakup metrik OSS Redis berikut.

Dengan pengecualian `ReplicationLag` dan `EngineCPUUtilization`, metrik ini berasal dari perintah Redis `OSSinfo`. Setiap metrik dihitung pada tingkat simpul.

Untuk dokumentasi lengkap OSS info perintah Redis, lihat <http://redis.io/commands/info>.

Lihat Juga


- [Metrik Tingkat Host](#)

Metrik	Deskripsi	Unit
ActiveDefragHits	Jumlah realokasi nilai per menit yang dilakukan oleh proses defragmentasi aktif. Ini berasal dari	Jumlah

Metrik	Deskripsi	Unit
	<code>active_defrag_hits</code> statistik di Redis OSS INFO .	
<code>AuthenticationFailures</code>	Jumlah total upaya gagal untuk mengautentikasi ke Redis OSS menggunakan perintah. AUTH Anda dapat menemukan informasi lebih lanjut tentang kegagalan otentikasi individu menggunakan ACLLOG perintah. Sebaiknya atur peringatan untuk hal ini guna mendeteksi percobaan akses yang tidak sah.	Hitung
	Jumlah total byte yang dialokasikan oleh MemoryDB untuk semua tujuan, termasuk dataset, buffer, dan sebagainya.	Byte
<code>BytesUsedForMemoryDB</code>	Dimension: Tier=SSD untuk cluster menggunakan Tingkatan data : Jumlah total byte yang digunakan oleh. SSD	Byte
	Dimension: Tier=Memory untuk cluster menggunakan Tingkatan data : Jumlah total byte yang digunakan oleh memori. Ini adalah nilai <code>used_memory</code> statistik di Redis OSS INFO .	Byte
<code>BytesReadFromDisk</code>	Jumlah total byte yang dibaca dari disk per menit. Didukung hanya untuk klaster yang menggunakan Tingkatan data .	Byte
<code>BytesWrittenToDisk</code>	Total jumlah byte yang ditulis ke disk per menit. Didukung hanya untuk klaster yang menggunakan Tingkatan data .	Byte

Metrik	Deskripsi	Unit
CommandAuthorizationFailures	Jumlah seluruh percobaan pengguna yang gagal untuk menjalankan perintah karena tidak memiliki izin untuk memanggil perintah itu. Anda dapat menemukan informasi lebih lanjut tentang kegagalan otentikasi individu menggunakan ACLLOG perintah. Sebaiknya atur peringatan untuk hal ini guna mendeteksi percobaan akses yang tidak sah.	Hitung
CurrConnections	Jumlah koneksi klien, tidak termasuk koneksi dari replika baca. MemoryDB menggunakan dua hingga empat koneksi untuk memantau cluster dalam setiap kasus. Ini berasal dari <code>connected_clients</code> statistik di Redis OSS INFO .	Hitung
CurrItems	Jumlah item dalam cache. Ini berasal dari OSS keyspace statistik Redis, menjumlahkan semua kunci di seluruh ruang kunci.	Hitung
	Dimension: Tier=Memory untuk klaster menggunakan Tingkatan data . Jumlah item dalam memori.	Hitungan
DatabaseMemoryUsagePercentage	Dimension: Tier=SSD (solid state drive) untuk klaster yang menggunakan Tingkatan data . Jumlah item di SSD.	Hitung
	Persentase memori yang tersedia untuk klaster yang sedang digunakan. Ini dihitung menggunakan <code>used_memory/maxmemory</code> dari Redis OSS INFO .	Persen

Metrik	Deskripsi	Unit
DatabaseCapacityUsagePercentage	<p>Persentase kapasitas data total untuk kluster yang sedang digunakan.</p> <p><u>Pada contoh Data Tiered, metrik dihitung sebagai $(\text{used_memory} - \text{mem_not_counted_for_evict} + \text{SSD used}) / (\text{maxmemory} + \text{SSD total capacity})$, di mana <code>used_memory</code> dan <code>maxmemory</code> diambil dari Redis. OSS INFO</u></p> <p>Dalam semua kasus lain, metrik dihitung menggunakan <code>used_memory/maxmemory</code>.</p>	Persen
DBAverageTTL	<p>Ekspos <code>avg_ttl</code> DBO dari keyspace statistik komando <u>Redis OSS INFO</u>.</p>	Milidetik

Metrik	Deskripsi	Unit
EngineCPUUtilization	<p>Menyediakan CPU pemanfaatan ulir OSS mesin Redis. Karena Redis OSS adalah single-threaded, Anda dapat menggunakan metrik ini untuk menganalisis beban proses Redis itu sendiri. OSS EngineCPU Utilization Metrik memberikan visibilitas yang lebih tepat dari proses RedisOSS. Anda dapat menggunakannya bersamaan dengan CPUUtilization metrik. CPUUtilization memaparkan CPU pemanfaatan untuk instance server secara keseluruhan, termasuk sistem operasi dan proses manajemen lainnya. Untuk tipe node yang lebih besar dengan empat vCPUs atau lebih, gunakan EngineCPU Utilization metrik untuk memantau dan menetapkan ambang batas untuk penskalaan.</p> <div data-bbox="594 1024 1268 1862"><p> Note</p><p>Pada host MemoryDB, proses latar belakang memantau host untuk memberikan pengalaman database terkelola. Proses latar belakang ini dapat mengambil sebagian besar CPU beban kerja. Ini tidak signifikan pada host yang lebih besar dengan lebih dari dua vCPUs. Tapi itu dapat mempengaruhi host yang lebih kecil dengan 2 vCPUs atau lebih sedikit. Jika Anda hanya memantau EngineCPU Utilization metrik, Anda tidak akan menyadari situasi di mana host kelebihan beban dengan CPU penggunaan tinggi dari Redis OSS dan CPU penggunaan tinggi dari proses</p></div>	Persen

Metrik	Deskripsi	Unit
	<p>pemantauan latar belakang. Oleh karena itu, kami merekomendasikan pemantauan CPUUtilization metrik untuk host dengan dua vCPUs atau kurang.</p>	
Evictions	Jumlah kunci yang telah dikosongkan karena batas maxmemory . Ini berasal dari evicted_keys statistik di Redis OSS INFO .	Hitung
IsPrimary	Menunjukkan apakah node adalah simpul utama dari pecahan saat ini. Metrik ini dapat bernilai 0 (bukan primer) atau 1 (primer).	Hitungan
KeyAuthorizationFailures	Jumlah seluruh percobaan pengguna yang gagal untuk mengakses kunci akses karena tidak mempunyai izin akses. Anda dapat menemukan informasi lebih lanjut tentang kegagalan otentikasi individu menggunakan ACLLOG perintah. Sebaiknya atur peringatan untuk hal ini guna mendeteksi percobaan akses yang tidak sah.	Hitung
KeyspaceHits	Jumlah pencarian kunci hanya-baca yang berhasil di kamus utama. Ini berasal dari keyspace_hits statistik di Redis OSS INFO .	Hitung
KeyspaceMisses	Jumlah pencarian kunci hanya-baca yang tidak berhasil di kamus utama. Ini berasal dari keyspace_misses statistik di Redis OSS INFO .	Hitung

Metrik	Deskripsi	Unit
KeysTracked	Jumlah kunci yang dilacak oleh pelacakan OSS kunci Redis sebagai persentase dari <code>tracking-table-max-keys</code> . Pelacakan kunci digunakan untuk membantu caching sisi klien dan memberitahukan klien jika kunci diubah.	Hitung
MaxReplicationThroughput	Throughput replikasi maksimum yang diamati selama siklus pengukuran terakhir.	Byte per detik
MemoryFragmentationRatio	Menunjukkan efisiensi dalam alokasi memori mesin RedisOSS. Ambang batas tertentu menandakan perilaku yang berbeda. Nilai yang disarankan adalah memiliki fragmentasi di atas 1,0. Ini dihitung <code>mem_fragmentation_ratio</code> statistik dari Redis OSS INFO .	Bilangan
NewConnections	Jumlah seluruh koneksi yang telah diterima oleh server selama periode ini. Ini berasal dari <code>total_connections_received</code> statistik di Redis OSS INFO .	Hitung
NumItemsReadFromDisk	Jumlah total item yang diambil dari disk per menit. Didukung hanya untuk klaster yang menggunakan Tingkatan data .	Hitungan
NumItemsWrittenToDisk	Jumlah total item yang ditulis ke disk per menit. Didukung hanya untuk klaster yang menggunakan Tingkatan data .	Hitungan
PrimaryLinkHealthStatus	Status ini memiliki dua nilai: 0 atau 1. Nilai 0 menunjukkan bahwa data di node utama MemoryDB tidak sinkron dengan OSS Redis aktif. EC2 Nilai 1 menunjukkan bahwa data sudah sinkron.	Boolean

Metrik	Deskripsi	Unit
Reclaimed	Jumlah seluruh peristiwa kedaluwarsa kunci. Ini berasal dari <code>expired_keys</code> statistik di Redis OSS INFO .	Hitung
ReplicationBytes	Untuk simpul dalam konfigurasi yang direplikasi, <code>ReplicationBytes</code> melaporkan jumlah byte yang dikirimkan oleh primer ke semua replikanya. Metrik ini mewakili beban tulis pada cluster. Ini berasal dari <code>master_repl_offset</code> statistik di Redis OSS INFO .	Byte
ReplicationDelayedWriteCommands	Jumlah perintah tulis yang tertunda karena replikasi sinkron. Replikasi dapat tertunda karena berbagai faktor, misalnya kemacetan jaringan atau melebihi throughput replikasi maksimum.	Hitung
ReplicationLag	Metrik ini hanya berlaku untuk simpul yang berjalan sebagai replika baca. Hal ini menunjukkan seberapa jauh ketinggalan, dalam detik, suatu replika dalam menerapkan perubahan dari simpul primer.	Detik

Berikut ini adalah kumpulan jenis perintah tertentu, yang berasal dari `info commandstats`. Bagian `commandstats` menyediakan statistik berdasarkan jenis perintah, termasuk jumlah panggilan.

Untuk daftar lengkap perintah yang tersedia, lihat perintah [Redis](#) dalam dokumentasi RedisOSS.

Metrik	Deskripsi	Unit
EvalBasedCmds	Jumlah seluruh perintah untuk perintah berbasis eval. Ini berasal dari <code>OSS commandstats</code> statistik Redis. Ini berasal dari <code>OSS commandstats</code> statistik Redis dengan menjumlahkan <code>eval</code> , <code>evalsha</code>	Hitung

Metrik	Deskripsi	Unit
GeoSpatialBasedCmds	Jumlah seluruh perintah untuk perintah berbasis geospasial. Ini berasal dari OSS <code>commandstats</code> statistik Redis. Ini diperoleh dengan menjumlahkan semua perintah jenis <code>geo</code> : <code>geoadd</code> , <code>geodist</code> , <code>geohash</code> , <code>geopos</code> , <code>georadius</code> , dan <code>georadiusbymember</code> .	Hitung
GetTypeCmds	Jumlah seluruh perintah jenis <code>read-only</code> . Ini berasal dari OSS <code>commandstats</code> statistik Redis dengan menjumlahkan semua perintah <code>read-only</code> tipe (<code>get</code> , <code>hget</code> , <code>scardlrange</code> , dan sebagainya.)	Hitung
HashBasedCmds	Jumlah seluruh perintah yang berbasis hash. Ini berasal dari OSS <code>commandstats</code> statistik Redis dengan menjumlahkan semua perintah yang bertindak atas satu atau lebih hash (<code>hget</code> , <code>hkeys</code> <code>hvalshdel</code> , dan seterusnya).	Hitung
HyperLogLogBasedCmds	Jumlah seluruh perintah berbasis <code>HyperLogLog</code> . Ini berasal dari OSS <code>commandstats</code> statistik Redis dengan menjumlahkan semua <code>pf</code> jenis perintah (<code>pfadd</code> , <code>pfcountpfmerge</code> , dan sebagainya.).	Hitung
JsonBasedCmds	Jumlah total perintah yang JSON berbasis. Ini berasal dari OSS <code>commandstats</code> statistik Redis dengan menjumlahkan semua perintah yang bertindak atas satu atau lebih objek JSON dokumen.	Hitung

Metrik	Deskripsi	Unit
KeyBasedCmds	Jumlah seluruh perintah yang berbasis kunci. Ini berasal dari OSS <code>commandstats</code> statistik Redis dengan menjumlahkan semua perintah yang bertindak atas satu atau lebih kunci di beberapa struktur data (<code>del</code> , <code>expire</code> , <code>rename</code> , dan seterusnya.).	Hitung
ListBasedCmds	Jumlah seluruh perintah yang berbasis daftar. Ini berasal dari OSS <code>commandstats</code> statistik Redis dengan menjumlahkan semua perintah yang bertindak atas satu atau lebih daftar (<code>lindex</code> , <code>lrange</code> , <code>lpushltrim</code> , dan seterusnya).	Hitung
PubSubBasedCmds	Jumlah seluruh perintah untuk fungsionalitas pub/sub. Ini berasal dari OSS <code>commandstats</code> statistik Redis dengan menjumlahkan semua perintah yang digunakan untuk fungsionalitas pub/sub: <code>psubscribe</code> , <code>publish</code> , <code>pubsub</code> , <code>punsubscribe</code> dan <code>unsubscribe</code> .	Hitung
SearchBasedCmds	Jumlah total indeks sekunder dan perintah pencarian, termasuk perintah baca dan tulis. Ini berasal dari OSS <code>commandstats</code> statistik Redis dengan menjumlahkan semua perintah pencarian yang bertindak atas indeks sekunder.	Hitung
SearchBasedGetCmds	Jumlah total indeks sekunder dan perintah pencarian hanya-baca. Ini berasal dari OSS <code>commandstats</code> statistik Redis dengan menjumlahkan semua perintah indeks sekunder dan pencarian <code>get</code> .	Hitung

Metrik	Deskripsi	Unit
SearchBasedSetCmds	Jumlah total indeks sekunder dan perintah tulis pencarian. Ini berasal dari OSS commandstats statistik Redis dengan menjumlahkan semua perintah indeks sekunder dan set pencarian.	Hitung
SearchNumberOfIndices	Jumlah total indeks.	Hitung
SearchNumberOfIndexedKeys	Jumlah total kunci Redis yang diindeks OSS	Hitung
SearchTotalIndexSize	Memori (byte) yang digunakan oleh semua indeks.	Byte
SetBasedCmds	Jumlah seluruh perintah yang berbasis set. Ini berasal dari OSS commandstats statistik Redis dengan menjumlahkan semua perintah yang bertindak atas satu atau lebih himpunan (scard,,sdiff, saddsunion, dan seterusnya).	Hitung
SetTypeCmds	Jumlah seluruh perintah jenis write. Ini berasal dari OSS commandstats statistik Redis dengan menjumlahkan semua mutative jenis perintah yang beroperasi pada data (set,,hset, saddlpop, dan sebagainya.)	Hitung
SortedSetBasedCmds	Jumlah seluruh perintah yang berbasis sorted set. Ini berasal dari OSS commandstats statistik Redis dengan menjumlahkan semua perintah yang bertindak atas satu atau lebih set yang diurutkan (zcount,, zrange zrankzadd, dan seterusnya).	Hitung

Metrik	Deskripsi	Unit
StringBasedCmds	Jumlah seluruh perintah yang berbasis string. Ini berasal dari OSS commandstats statistik Redis dengan menjumlahkan semua perintah yang bertindak atas satu atau lebih string (strlen,, setexsetrange, dan seterusnya).	Hitung
StreamBasedCmds	Jumlah seluruh perintah yang berbasis aliran. Ini berasal dari OSS commandstats statistik Redis dengan menjumlahkan semua perintah yang bertindak atas satu atau lebih jenis data aliran (xrange,, xlen xaddxdel, dan seterusnya).	Hitung

Metrik Apa yang Harus Saya Pantau?

CloudWatch Metrik berikut menawarkan wawasan yang baik tentang kinerja MemoryDB. Dalam kebanyakan kasus, kami menyarankan Anda menyetel CloudWatch alarm untuk metrik ini sehingga Anda dapat mengambil tindakan korektif sebelum masalah kinerja terjadi.

Metrik yang Perlu Dipantau

- [CPUUtilization](#)
- [EngineCPUUtilization](#)
- [SwapUsage](#)
- [Evictions](#)
- [CurrConnections](#)
- [Memori](#)
- [Jaringan](#)
- [Replikasi](#)

CPUUtilization

Ini adalah metrik tingkat host yang dilaporkan sebagai persentase. Untuk informasi selengkapnya, lihat [Metrik Tingkat Host](#).

Untuk tipe node yang lebih kecil dengan 2 vCPUs atau kurang, gunakan `CPUUtilization` metrik untuk memantau beban kerja Anda.

Secara umum, kami sarankan Anda menetapkan ambang batas Anda pada 90% dari yang tersedia CPU. Karena Redis OSS adalah single-threaded, nilai ambang sebenarnya harus dihitung sebagai sebagian kecil dari total kapasitas node. Sebagai contoh, misalkan Anda menggunakan jenis simpul yang memiliki dua inti. Dalam hal ini, ambang batas untuk `CPUUtilization` adalah $90/2$, atau 45%. Untuk menemukan jumlah core (vCPUs) yang dimiliki tipe node Anda, lihat Harga [MemoryDB](#).

Anda perlu menentukan ambang batas Anda sendiri, berdasarkan jumlah core di node yang Anda gunakan. Jika Anda melebihi ambang batas ini, dan beban kerja utama Anda berasal dari permintaan baca, skala klaster Anda dengan menambahkan replika baca. Jika beban kerja utama berasal dari permintaan tulis, kami sarankan Anda menambahkan lebih banyak pecahan untuk mendistribusikan beban kerja tulis di lebih banyak node utama.

Tip

Alih-alih menggunakan metrik `Host-LevelCPUUtilization`, Anda mungkin dapat menggunakan OSS metrik `RedisEngineCPUUtilization`, yang melaporkan persentase penggunaan pada inti mesin OSS Redis. Untuk melihat apakah metrik ini tersedia di node Anda dan untuk informasi selengkapnya, lihat [Metrik untuk MemoryDB](#).

Untuk tipe node yang lebih besar dengan 4 vCPUs atau lebih, Anda mungkin ingin menggunakan `EngineCPUUtilization` metrik, yang melaporkan persentase penggunaan pada inti OSS mesin Redis. Untuk melihat apakah metrik ini tersedia di node Anda dan untuk informasi selengkapnya, lihat [Metrik untuk MemoryDB](#).

EngineCPUUtilization

Untuk tipe node yang lebih besar dengan 4 vCPUs atau lebih, Anda mungkin ingin menggunakan `EngineCPUUtilization` metrik, yang melaporkan persentase penggunaan pada inti OSS mesin Redis. Untuk melihat apakah metrik ini tersedia di node Anda dan untuk informasi selengkapnya, lihat [Metrik untuk MemoryDB](#).

SwapUsage

Ini adalah metrik tingkat host yang dilaporkan dalam byte. Untuk informasi selengkapnya, lihat [Metrik Tingkat Host](#).

Metrik ini tidak boleh melebihi 50 MB.

Evictions

Ini adalah metrik mesin. Sebaiknya tentukan ambang batas alarm Anda sendiri untuk metrik ini berdasarkan kebutuhan aplikasi Anda.

CurrConnections

Ini adalah metrik mesin. Sebaiknya tentukan ambang batas alarm Anda sendiri untuk metrik ini berdasarkan kebutuhan aplikasi Anda.

Peningkatan jumlah `CurrConnections` mungkin menunjukkan masalah dengan aplikasi Anda; Anda perlu menyelidiki perilaku aplikasi untuk mengatasi masalah ini.

Memori

Memori adalah aspek inti dari RedisOSS. Memahami pemanfaatan memori dari klaster Anda diperlukan untuk menghindari kehilangan data dan mengakomodasi pertumbuhan set data Anda pada masa mendatang. Statistik tentang pemanfaatan memori node tersedia di bagian memori dari perintah Redis OSS [INFO](#).

Jaringan

Salah satu faktor penentu untuk kapasitas bandwidth jaringan dari klaster Anda adalah jenis simpul yang telah Anda pilih. Untuk informasi selengkapnya tentang kapasitas jaringan node Anda, lihat [harga Amazon MemoryDB](#).

Replikasi

Volume data yang direplikasi akan terlihat melalui metrik `ReplicationBytes`. Anda dapat memantau `MaxReplicationThroughput` terhadap throughput kapasitas replikasi. Disarankan untuk menambahkan lebih banyak pecahan saat mencapai throughput kapasitas replikasi maksimum.

`ReplicationDelayedWriteCommands` juga dapat menunjukkan apakah beban kerja melebihi throughput kapasitas replikasi maksimum. [Untuk informasi lebih lanjut tentang replikasi di MemoryDB, lihat Memahami replikasi MemoryDB](#)

Memilih Statistik dan Periode Metrik

Meskipun CloudWatch akan memungkinkan Anda untuk memilih statistik dan periode untuk setiap metrik, tidak semua kombinasi akan berguna. Misalnya, statistik Rata-rata, Minimum, dan Maksimum berguna, tetapi statistik Jumlah tidak. CPUUtilization

Semua sampel MemoryDB diterbitkan untuk durasi 60 detik untuk setiap node individu. Untuk periode 60 detik, metrik simpul hanya akan berisi satu sampel.

CloudWatch Metrik pemantauan

MemoryDB dan CloudWatch terintegrasi sehingga Anda dapat mengumpulkan berbagai metrik. Anda dapat memantau metrik ini menggunakan CloudWatch.

Note

Contoh berikut memerlukan alat baris CloudWatch perintah. Untuk informasi selengkapnya tentang CloudWatch dan untuk mengunduh alat pengembang, lihat [halaman CloudWatch produk](#).

Prosedur berikut menunjukkan cara menggunakan CloudWatch untuk mengumpulkan statistik ruang penyimpanan untuk sebuah cluster selama satu jam terakhir.

Note

Nilai `StartTime` dan `EndTime` nilai yang diberikan dalam contoh berikut adalah untuk tujuan ilustrasi. Pastikan untuk mengganti nilai waktu mulai dan akhir yang sesuai untuk node Anda.

Untuk informasi tentang batas MemoryDB, lihat [batas AWS layanan](#) untuk MemoryDB.

CloudWatch Metrik pemantauan (Konsol)

Untuk mengumpulkan statistik CPU pemanfaatan untuk sebuah cluster

1. Masuk ke AWS Management Console dan buka konsol MemoryDB di. <https://console.aws.amazon.com/memorydb/>
2. Pilih node yang ingin Anda lihat metrik.

Note

Memilih lebih dari 20 simpul akan menonaktifkan tampilan metrik pada konsol.

- a. Pada halaman Clusters dari AWS Management Console, klik nama satu atau beberapa cluster.

Halaman detail untuk cluster muncul.

- b. Klik tab Simpul di bagian atas jendela.
- c. Pada tab Nodes pada jendela detail, pilih node yang ingin Anda lihat metrik.

Daftar CloudWatch Metrik yang tersedia muncul di bagian bawah jendela konsol.

- d. Klik pada metrik CPU Pemanfaatan.

CloudWatch Konsol akan terbuka, menampilkan metrik yang Anda pilih. Anda dapat menggunakan kotak daftar drop-down Statistik dan Periode, serta tab Rentang waktu untuk mengubah metrik yang ditampilkan.

CloudWatch Metrik pemantauan menggunakan CloudWatch CLI

Untuk mengumpulkan statistik CPU pemanfaatan untuk sebuah cluster

- Gunakan CloudWatch perintah `aws cloudwatch get-metric-statistics` dengan parameter berikut (perhatikan bahwa waktu mulai dan akhir ditampilkan sebagai contoh saja; Anda harus mengganti waktu mulai dan akhir yang sesuai):

Untuk Linux, macOS, atau Unix:

```
aws cloudwatch get-metric-statistics CPUUtilization \  
  --dimensions=ClusterName=mycluster,NodeId=0002 \  
  --statistics=Average \  
  --namespace="AWS/MemoryDB" \  
  --start-time 2013-07-05T00:00:00 \  
  --end-time 2013-07-06T00:00:00 \  
  --period=60
```

Untuk Windows:

```
mon-get-stats CPUUtilization ^
  --dimensions=ClusterName=mycluster,NodeId=0002" ^
  --statistics=Average ^
  --namespace="AWS/MemoryDB" ^
  --start-time 2013-07-05T00:00:00 ^
  --end-time 2013-07-06T00:00:00 ^
  --period=60
```

CloudWatch Metrik pemantauan menggunakan CloudWatch API

Untuk mengumpulkan statistik CPU pemanfaatan untuk sebuah cluster

- Panggil CloudWatch API `GetMetricStatistics` dengan parameter berikut (perhatikan bahwa waktu mulai dan akhir ditampilkan sebagai contoh saja; Anda perlu mengganti waktu mulai dan akhir yang sesuai):
 - `Statistics.member.1=Average`
 - `Namespace=AWS/MemoryDB`
 - `StartTime=2013-07-05T00:00:00`
 - `EndTime=2013-07-06T00:00:00`
 - `Period=60`
 - `MeasureName=CPUUtilization`
 - `Dimensions=ClusterName=mycluster,NodeId=0002`

Example

```
http://monitoring.amazonaws.com/
  ?SignatureVersion=4
  &Action=GetMetricStatistics
  &Version=2014-12-01
  &StartTime=2013-07-16T00:00:00
  &EndTime=2013-07-16T00:02:00
  &Period=60
  &Statistics.member.1=Average
  &Dimensions.member.1="ClusterName=mycluster"
  &Dimensions.member.2="NodeId=0002"
```

```
&Namespace=Amazon/memorydb
&MeasureName=CPUUtilization
&Timestamp=2013-07-07T17%3A48%3A21.746Z
&AWS;AccessKeyId=<&AWS; Access Key ID>
&Signature=<Signature>
```

Memantau acara MemoryDB

Ketika peristiwa penting terjadi untuk sebuah cluster, MemoryDB mengirimkan pemberitahuan ke topik Amazon SNS tertentu. Contohnya meliputi kegagalan menambahkan simpul, keberhasilan menambahkan simpul, perubahan grup keamanan, dan lainnya. Dengan memantau peristiwa penting, Anda dapat mengetahui status klaster Anda saat ini dan dapat mengambil tindakan korektif sesuai peristiwa tersebut.

Topik

- [Mengelola pemberitahuan Amazon MemoryDB SNS](#)
- [Melihat acara MemoryDB](#)
- [Notifikasi Peristiwa dan Amazon SNS](#)

Mengelola pemberitahuan Amazon MemoryDB SNS

Anda dapat mengonfigurasi MemoryDB untuk mengirim pemberitahuan untuk peristiwa klaster penting menggunakan Amazon Simple Notification Service (AmazonSNS). Dalam contoh ini, Anda akan mengonfigurasi cluster dengan Amazon Resource Name (ARN) dari SNS topik Amazon untuk menerima pemberitahuan.

Note

Topik ini mengasumsikan bahwa Anda telah mendaftar ke Amazon SNS dan telah menyiapkan serta berlangganan topik AmazonSNS. Untuk informasi selengkapnya tentang cara melakukannya, lihat [Panduan Developer Amazon Simple Notification Service](#).

Menambahkan SNS topik Amazon

Bagian berikut menunjukkan cara menambahkan SNS topik Amazon menggunakan AWS Konsol, the AWS CLI, atau API MemoryDB.

Menambahkan SNS topik Amazon (Konsol)

Prosedur berikut menunjukkan cara menambahkan SNS topik Amazon untuk cluster.

Note

Proses ini juga dapat digunakan untuk memodifikasi SNS topik Amazon.

Untuk menambah atau memodifikasi SNS topik Amazon untuk klaster (Konsol)

1. Masuk ke AWS Management Console dan buka konsol MemoryDB di <https://console.aws.amazon.com/memorydb/>
2. Di Cluster, pilih klaster yang ingin Anda tambahkan atau ubah SNS topik ARN Amazon.
3. Pilih Ubah.
4. Di Ubah Kluster di bawah Topik untuk SNS Pemberitahuan, pilih SNS topik yang ingin Anda tambahkan, atau pilih ARNInput manual dan ketik SNS topik Amazon. ARN
5. Pilih Ubah.

Menambahkan SNS topik Amazon (AWS CLI)

Untuk menambahkan atau memodifikasi SNS topik Amazon untuk klaster, gunakan AWS CLI perintah `update-cluster`.

Contoh kode berikut menambahkan SNS topik Amazon arn ke my-cluster.

Untuk Linux, macOS, atau Unix:

```
aws memorydb update-cluster \  
  --cluster-name my-cluster \  
  --sns-topic-arn arn:aws:sns:us-east-1:565419523791:memorydbNotifications
```

Untuk Windows:

```
aws memorydb update-cluster ^  
  --cluster-name my-cluster ^  
  --sns-topic-arn arn:aws:sns:us-east-1:565419523791:memorydbNotifications
```

Untuk informasi lebih lanjut, lihat [UpdateCluster](#).

Menambahkan SNS topik Amazon (APIMemoryDB)

Untuk menambah atau memperbarui SNS topik Amazon untuk klaster, panggil `UpdateCluster` tindakan dengan parameter berikut:

- `ClusterName=my-cluster`
- `SnsTopicArn=arn%3Aaws%3Asns%3Aus-east-1%3A565419523791%3AmemorydbNotifications`

Untuk menambahkan atau memperbarui SNS topik Amazon untuk klaster, panggil `UpdateCluster` tindakan.

Untuk informasi lebih lanjut, lihat [UpdateCluster](#).

Mengaktifkan dan menonaktifkan notifikasi Amazon SNS

Anda dapat mengaktifkan atau menonaktifkan notifikasi untuk klaster. Prosedur berikut menunjukkan cara menonaktifkan SNS notifikasi Amazon.

Mengaktifkan dan menonaktifkan notifikasi SNS Amazon (Konsol)

Untuk menonaktifkan SNS notifikasi Amazon menggunakan AWS Management Console

1. Masuk ke AWS Management Console dan buka konsol MemoryDB di <https://console.aws.amazon.com/memorydb/>
2. Pilih tombol radio di sebelah kiri cluster yang ingin Anda ubah notifikasi.
3. Pilih Ubah.
4. Di Modify Cluster di bawah Topik untuk SNS Pemberitahuan, pilih Nonaktifkan Pemberitahuan.
5. Pilih Ubah.

Mengaktifkan dan menonaktifkan notifikasi SNS Amazon (AWS CLI)

Untuk menonaktifkan SNS notifikasi Amazon, gunakan perintah `update-cluster` dengan parameter berikut:

Untuk Linux, macOS, atau Unix:

```
aws memorydb update-cluster \  
  --cluster-name my-cluster \  
  --sns-topic-arn arn:aws:sns:us-east-1:565419523791:memorydb-notifications
```

```
--sns-topic-status inactive
```

Untuk Windows:

```
aws memorydb update-cluster ^  
  --cluster-name my-cluster ^  
  --sns-topic-status inactive
```

Mengaktifkan dan menonaktifkan pemberitahuan SNS Amazon (MemoryDB) API

Untuk menonaktifkan SNS notifikasi Amazon, panggil UpdateCluster tindakan dengan parameter berikut:

- ClusterName=my-cluster
- SnsTopicStatus=inactive

Panggilan ini menghasilkan output seperti yang berikut ini:

Example

```
https://memory-db.us-east-1.amazonaws.com/  
  ?Action=UpdateCluster  
  &ClusterName=my-cluster  
  &SnsTopicStatus=inactive  
  &Version=2021-01-01  
  &SignatureVersion=4  
  &SignatureMethod=HmacSHA256  
  &Timestamp=20210801T220302Z  
  &X-Amz-Algorithm=Amazon4-HMAC-SHA256  
  &X-Amz-Date=20210801T220302Z  
  &X-Amz-SignedHeaders=Host  
  &X-Amz-Expires=20210801T220302Z  
  &X-Amz-Credential=<credential>  
  &X-Amz-Signature=<signature>
```

Melihat acara MemoryDB

MemoryDB mencatat peristiwa yang berhubungan dengan cluster, grup keamanan, dan grup parameter Anda. Informasi ini mencakup tanggal dan waktu peristiwa, nama sumber dan jenis sumber peristiwa, serta deskripsi dari peristiwa tersebut. Anda dapat dengan mudah mengambil peristiwa dari log menggunakan konsol MemoryDB, AWS CLI `describe-events` perintah, atau tindakan MemoryDB. API `DescribeEvents`

Prosedur berikut menunjukkan cara melihat semua acara MemoryDB selama 24 jam terakhir (1440 menit).

Melihat acara MemoryDB (Konsol)

Prosedur berikut menampilkan peristiwa menggunakan konsol MemoryDB.

Untuk melihat acara menggunakan konsol MemoryDB

1. Masuk ke AWS Management Console dan buka konsol MemoryDB di <https://console.aws.amazon.com/memorydb/>
2. Di panel navigasi kiri, pilih Acara.

Layar Acara muncul mencantumkan semua acara yang tersedia. Setiap baris daftar mewakili satu peristiwa dan menampilkan sumber peristiwa, jenis acara (seperti cluster, parameter-grup, acl, security-group atau subnet group), GMT waktu acara, dan deskripsi acara.

Dengan Filter, Anda dapat menentukan apakah Anda ingin melihat semua peristiwa, atau hanya peristiwa dari jenis tertentu dalam daftar peristiwa.

Melihat acara MemoryDB (AWS CLI)

Untuk menghasilkan daftar peristiwa MemoryDB menggunakan AWS CLI, gunakan perintah `describe-events` Anda dapat menggunakan parameter opsional untuk mengontrol jenis peristiwa yang tercantum, jangka waktu peristiwa yang dicantumkan, jumlah maksimum peristiwa yang akan dicantumkan, dan lainnya.

Kode berikut mencantumkan hingga 40 peristiwa cluster.

```
aws memorydb describe-events --source-type cluster --max-results 40
```

Kode berikut menampilkan daftar semua peristiwa selama 24 jam terakhir (1.440 menit).

```
aws memorydb describe-events --duration 1440
```

Output dari perintah `describe-events` terlihat seperti berikut ini.

```
{
  "Events": [
    {
      "Date": "2021-03-29T22:17:37.781Z",
      "Message": "Added node 0001 in Availability Zone us-east-1a",
      "SourceName": "memorydb01",
      "SourceType": "cluster"
    },
    {
      "Date": "2021-03-29T22:17:37.769Z",
      "Message": "cluster created",
      "SourceName": "memorydb01",
      "SourceType": "cluster"
    }
  ]
}
```

Untuk informasi selengkapnya, seperti parameter yang tersedia dan nilai parameter yang diizinkan, lihat [describe-events](#).

Melihat acara MemoryDB (MemoryDB) API

Untuk menghasilkan daftar acara MemoryDB menggunakan MemoryDBAPI, gunakan tindakan `DescribeEvents` Anda dapat menggunakan parameter opsional untuk mengontrol jenis peristiwa yang tercantum, jangka waktu peristiwa yang dicantumkan, jumlah maksimum peristiwa yang akan dicantumkan, dan lainnya.

Kode berikut mencantumkan 40 peristiwa `-cluster` terbaru.

```
https://memory-db.us-east-1.amazonaws.com/
?Action=DescribeEvents
&MaxResults=40
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&SourceType=cluster
&Timestamp=20210802T192317Z
&Version=2021-01-01
```

```
&X-Amz-Credential=<credential>
```

Kode berikut mencantumkan peristiwa cluster selama 24 jam terakhir (1440 menit).

```
https://memory-db.us-east-1.amazonaws.com/  
?Action=DescribeEvents  
&Duration=1440  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&SourceType=cluster  
&Timestamp=20210802T192317Z  
&Version=2021-01-01  
&X-Amz-Credential=<credential>
```

Perintah di atas akan menghasilkan output yang serupa dengan berikut ini.

```
<DescribeEventsResponse xmlns="http://memory-db.us-east-1.amazonaws.com/  
doc/2021-01-01/">  
  <DescribeEventsResult>  
    <Events>  
      <Event>  
        <Message>cluster created</Message>  
        <SourceType>cluster</SourceType>  
        <Date>2021-08-02T18:22:18.202Z</Date>  
        <SourceName>my-memorydb-primary</SourceName>  
      </Event>  
  
      (...output omitted...)  
  
    </Events>  
  </DescribeEventsResult>  
  <ResponseMetadata>  
    <RequestId>e21c81b4-b9cd-11e3-8a16-7978bb24ffdf</RequestId>  
  </ResponseMetadata>  
</DescribeEventsResponse>
```

Untuk informasi selengkapnya, seperti parameter yang tersedia dan nilai parameter yang diizinkan, lihat [DescribeEvents](#).

Notifikasi Peristiwa dan Amazon SNS

MemoryDB dapat mempublikasikan pesan menggunakan Amazon Simple Notification Service (SNS) ketika peristiwa penting terjadi di klaster. Fitur ini dapat digunakan untuk menyegarkan daftar server pada mesin klien yang terhubung ke titik akhir node individual dari sebuah cluster.

Note

Untuk informasi selengkapnya tentang Amazon Simple Notification Service (SNS), termasuk informasi tentang harga dan tautan ke dokumentasi Amazon SNS, lihat [Halaman produk Amazon SNS](#).

Notifikasi dipublikasikan ke topik Amazon SNS tertentu. Berikut ini adalah persyaratan untuk notifikasi:

- Hanya satu topik yang dapat dikonfigurasi untuk pemberitahuan MemoryDB.
- AWS Akun yang memiliki topik Amazon SNS harus merupakan akun yang sama yang memiliki cluster tempat notifikasi diaktifkan.


Acara MemoryDB


Peristiwa MemoryDB berikut memicu notifikasi Amazon SNS:

Nama Peristiwa	Pesan	Deskripsi
MemoryDB: AddNodeComplete	"Modified number of nodes from %d to %d"	Sebuah node telah ditambahkan ke cluster dan siap digunakan.
MemoryDB: AddNodeFailed karena alamat IP gratis yang tidak mencukupi	"Failed to modify number of nodes from %d to %d due to insufficient free IP addresses"	Sebuah node tidak dapat ditambahkan karena tidak ada cukup alamat IP yang tersedia.
MemoryDB: ClusterParametersChanged	"Updated parameter group for the cluster"	Satu atau lebih parameter cluster telah diubah.

Nama Peristiwa	Pesan	Deskripsi
	Dalam hal membuat, kirim juga "Updated to use a ParameterGroup %s"	
MemoryDB: ClusterProvisioningComplete	"Cluster created."	Penyediaan cluster selesai, dan node di cluster siap digunakan.
MemoryDB: ClusterProvisioningFailed karena status jaringan yang tidak kompatibel	"Failed to create cluster due to incompatible network state. %s"	Upaya dilakukan untuk meluncurkan cluster baru ke virtual private cloud (VPC) yang tidak ada.
MemoryDB: ClusterRestoreFailed	"Restore from %s failed for node %s. %s"	<p>MemoryDB tidak dapat mengisi cluster dengan data snapshot Redis OSS. Ini mungkin karena file snapshot yang tidak ada di Amazon S3, atau izin yang salah pada file itu. Jika Anda menggambar cluster, statusnya akan <code>restore-failed</code>. Anda harus menghapus cluster dan memulai dari awal.</p> <p>Untuk informasi selengkapnya, lihat Menyemai cluster baru dengan snapshot yang dibuat secara eksternal.</p>
MemoryDB: ClusterScalingComplete	"Succeeded applying modification to node type to %s."	Skala untuk klaster selesai dengan sukses.

Nama Peristiwa	Pesan	Deskripsi
MemoryDB: ClusterScalingFailed	"Failed applying modification to node type to %s."	Operasi penskalaan pada cluster gagal.
MemoryDB: ClusterSecurityGroupModified	"Modified security group for cluster."	Salah satu peristiwa berikut telah terjadi: <ul style="list-style-type: none">• Daftar grup keamanan yang diotorisasi untuk cluster telah dimodifikasi.• Satu atau lebih grup keamanan EC2 baru telah diotorisasi pada salah satu grup keamanan yang terkait dengan cluster.• Satu atau lebih kelompok keamanan EC2 telah dicabut dari salah satu kelompok keamanan yang terkait dengan cluster.

Nama Peristiwa	Pesan	Deskripsi
MemoryDB: NodeReplacementStarted	"Recovering node %s"	<p>MemoryDB telah mendeteksi bahwa host yang menjalankan node terdegradasi atau tidak dapat dijangkau dan telah mulai mengganti node.</p> <div data-bbox="1068 495 1510 760"><p> Note</p><p>Entri DNS untuk node yang diganti tidak berubah.</p></div> <p>Pada kebanyakan kasus, Anda tidak perlu menyegarkan daftar server untuk klien Anda ketika peristiwa ini terjadi. Namun, beberapa pustaka klien mungkin berhenti menggunakan node bahkan setelah MemoryDB mengganti node; dalam hal ini, aplikasi harus menyegarkan daftar server ketika peristiwa ini terjadi.</p>

Nama Peristiwa	Pesan	Deskripsi
MemoryDB: NodeRepl ceComplete	"Finished recovery for node %s"	<p>MemoryDB telah mendeteksi bahwa host yang menjalankan node terdegradasi atau tidak dapat dijangkau dan telah selesai mengganti node.</p> <div data-bbox="1068 493 1507 760" style="border: 1px solid #add8e6; border-radius: 10px; padding: 10px;"><p> Note Entri DNS untuk node yang diganti tidak berubah.</p></div> <p>Pada kebanyakan kasus, Anda tidak perlu menyegarkan daftar server untuk klien Anda ketika peristiwa ini terjadi. Namun, beberapa pustaka klien mungkin berhenti menggunakan node bahkan setelah MemoryDB mengganti node; dalam hal ini, aplikasi harus menyegarkan daftar server ketika peristiwa ini terjadi.</p>
MemoryDB: CreateClu sterComplete	"Cluster created"	Cluster berhasil dibuat.

Nama Peristiwa	Pesan	Deskripsi
MemoryDB: CreateClusterFailed	"Failed to create cluster due to unsuccessful creation of its node(s)." dan "Deleting all nodes belonging to this cluster."	Cluster tidak dibuat.
MemoryDB: DeleteClusterComplete	"Cluster deleted."	Penghapusan cluster dan semua node terkait telah selesai.
MemoryDB: FailoverComplete	"Failover to replica node %s completed"	Failover ke simpul replika telah berhasil.
MemoryDB: NodeReplacementCanceled	"The replacement of node %s which was scheduled during the maintenance window from start time: %s, end time: %s has been canceled"	Sebuah simpul di klaster Anda yang dijadwalkan akan diganti tidak lagi dijadwalkan akan diganti.
MemoryDB: NodeReplacementRescheduled	"The replacement in maintenance window for node %s has been re-scheduled from previous start time: %s, previous end time: %s to new start time: %s, new end time: %s"	Sebuah simpul di klaster Anda yang sebelumnya dijadwalkan akan diganti telah dijadwalkan ulang untuk diganti selama periode baru yang dijelaskan dalam notifikasi. Untuk informasi tentang tindakan yang dapat Anda lakukan, lihat Mengganti simpul .

Nama Peristiwa	Pesan	Deskripsi
MemoryDB: NodeReplacementScheduled	"The node %s is scheduled for replacement during the maintenance window from start time: %s to end time: %s"	Sebuah simpul di kluster Anda dijadwalkan untuk penggantian selama jendela yang dijelaskan dalam notifikasi. Untuk informasi tentang tindakan yang dapat Anda ambil, lihat Mengganti simpul .
MemoryDB: RemoveNodeComplete	"Removed node %s"	Sebuah node telah dihapus dari cluster.
MemoryDB: SnapshotComplete	"Snapshot %s succeeded for node %s"	Sebuah snapshot telah berhasil diselesaikan.
MemoryDB: SnapshotFailed	"Snapshot %s failed for node %s"	Sebuah snapshot telah gagal. Lihat peristiwa cluster untuk penyebab lebih rinci. Jika Anda menggambarkan snapshot, lihat DescribeSnapshots , statusnya akan failed.

Mencatat panggilan API MemoryDB dengan AWS CloudTrail

MemoryDB terintegrasi dengan AWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan di MemoryDB. CloudTrail menangkap semua panggilan API untuk MemoryDB sebagai peristiwa, termasuk panggilan dari konsol MemoryDB dan dari panggilan kode ke operasi API MemoryDB. Jika Anda membuat jejak, Anda dapat mengaktifkan pengiriman CloudTrail acara secara terus menerus ke bucket Amazon S3, termasuk acara untuk MemoryDB. Jika Anda tidak mengonfigurasi jejak, Anda masih dapat melihat peristiwa terbaru di CloudTrail konsol dalam Riwayat acara. Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat ke MemoryDB, alamat IP dari mana permintaan dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan.

Untuk mempelajari selengkapnya CloudTrail, lihat [Panduan AWS CloudTrail Pengguna](#).

Informasi MemoryDB di CloudTrail

CloudTrail diaktifkan di AWS akun Anda saat Anda membuat akun. Ketika aktivitas terjadi di MemoryDB, aktivitas tersebut direkam dalam suatu CloudTrail peristiwa bersama dengan peristiwa AWS layanan lainnya dalam riwayat Acara. Anda dapat melihat, mencari, dan mengunduh acara terbaru di AWS akun Anda. Untuk informasi selengkapnya, lihat [Melihat Acara dengan Riwayat CloudTrail Acara](#).

Untuk catatan peristiwa yang sedang berlangsung di AWS akun Anda, termasuk acara untuk MemoryDB, buat jejak. Jejak memungkinkan CloudTrail untuk mengirimkan file log ke bucket Amazon S3. Secara default, ketika Anda membuat jejak di konsol, jejak ini diterapkan ke semua Wilayah. Trail mencatat peristiwa dari semua wilayah di AWS partisi dan mengirimkan file log ke bucket Amazon S3 yang Anda tentukan. Selain itu, Anda dapat mengonfigurasi AWS layanan lain untuk menganalisis lebih lanjut dan menindaklanjuti data peristiwa yang dikumpulkan dalam CloudTrail log. Untuk informasi selengkapnya, lihat berikut:

- [Gambaran Umum untuk Membuat Jejak](#)
- [CloudTrail Layanan dan Integrasi yang Didukung](#)
- [Mengonfigurasi Notifikasi Amazon SNS untuk CloudTrail](#)
- [Menerima File CloudTrail Log dari Beberapa Wilayah](#) dan [Menerima File CloudTrail Log dari Beberapa Akun](#)

Semua tindakan MemoryDB dicatat oleh CloudTrail. Misalnya, panggilan ke `CreateCluster`, `DescribeClusters` dan `UpdateCluster` tindakan menghasilkan entri dalam file CloudTrail log.

Setiap entri peristiwa atau log berisi informasi tentang siapa yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan hal berikut ini:

- Apakah permintaan tersebut dibuat dengan kredensial root atau pengguna IAM.
- Apakah permintaan tersebut dibuat dengan kredensial keamanan sementara untuk satu peran atau pengguna terfederasi.
- Apakah permintaan itu dibuat oleh AWS layanan lain.

Untuk informasi selengkapnya, lihat Elemen [CloudTrail UserIdentity](#).

Memahami entri file log MemoryDB

Trail adalah konfigurasi yang memungkinkan pengiriman peristiwa sebagai file log ke bucket Amazon S3 yang Anda tentukan. CloudTrail file log berisi satu atau lebih entri log. Peristiwa mewakili permintaan tunggal dari sumber manapun dan mencakup informasi tentang tindakan yang diminta, tanggal dan waktu tindakan, parameter permintaan, dan sebagainya. CloudTrail file log bukanlah jejak tumpukan yang diurutkan dari panggilan API publik, sehingga file tersebut tidak muncul dalam urutan tertentu.

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan `CreateCluster` tindakan.

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EKIAUAXQT3SWDEXAMPLE",
    "arn": "arn:aws:iam::123456789012:user/john",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "john"
  },
  "eventTime": "2021-07-10T17:56:46Z",
  "eventSource": "memorydb.amazonaws.com",
  "eventName": "CreateCluster",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.01",
  "userAgent": "aws-cli/2.2.29 Python/3.9.6 Darwin/19.6.0 source/x86_64 prompt/off
command/memorydb.create-cluster",
  "requestParameters": {
    "clusterName": "memorydb-cluster",
    "nodeType": "db.r6g.large",
    "subnetGroupName": "memorydb-subnet-group",
    "aCLName": "open-access"
  },
  "responseElements": {
    "cluster": {
      "name": "memorydb-cluster",
      "status": "creating",
      "numberOfShards": 1,
      "availabilityMode": "MultiAZ",
      "clusterEndpoint": {
        "port": 6379
      }
    }
  },
}
```

```

        "nodeType": "db.r6g.large",
        "engineVersion": "6.2",
        "enginePatchVersion": "6.2.6",
        "parameterGroupName": "default.memorydb-redis6",
        "parameterGroupStatus": "in-sync",
        "subnetGroupName": "memorydb-subnet-group",
        "tLSEnabled": true,
        "aRN": "arn:aws:memorydb:us-east-1:123456789012:cluster/memorydb-cluster",
        "snapshotRetentionLimit": 0,
        "maintenanceWindow": "tue:06:30-tue:07:30",
        "snapshotWindow": "09:00-10:00",
        "aCLName": "open-access",
        "dataTiering": "false",
        "autoMinorVersionUpgrade": true
    }
},
"requestID": "506fc951-9ae2-42bb-872c-98028dc8ed11",
"eventID": "2ecf3dc3-c931-4df0-a2b3-be90b596697e",
"readOnly": false,
"eventType": "AwsApiCall",
"managementEvent": true,
"recipientAccountId": "123456789012",
"eventCategory": "Management"
}

```

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan DescribeClusters tindakan. Perhatikan bahwa untuk semua panggilan MemoryDB Describe dan List (Describe* dan List*), responseElements bagian dihapus dan muncul sebagai null

```

{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "EKIAUAXQT3SWDEXAMPLE",
    "arn": "arn:aws:iam::123456789012:user/john",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "john"
  },
  "eventTime": "2021-07-10T18:39:51Z",
  "eventSource": "memorydb.amazonaws.com",
  "eventName": "DescribeClusters",
  "awsRegion": "us-east-1",

```

```

    "sourceIPAddress": "192.0.2.01",
    "userAgent": "aws-cli/2.2.29 Python/3.9.6 Darwin/19.6.0 source/x86_64 prompt/off
command/memorydb.describe-clusters",
    "requestParameters": {
        "maxResults": 50,
        "showShardDetails": true
    },
    "responseElements": null,
    "requestID": "5e831993-52bb-494d-9bba-338a117c2389",
    "eventID": "32a3dc0a-31c8-4218-b889-1a6310b7dd50",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management"
}

```

Contoh berikut menunjukkan entri CloudTrail log yang mencatat UpdateCluster tindakan.

```

{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EKIAUAXQT3SWDEXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/john",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "john"
    },
    "eventTime": "2021-07-10T19:23:20Z",
    "eventSource": "memorydb.amazonaws.com",
    "eventName": "UpdateCluster",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.01",
    "userAgent": "aws-cli/2.2.29 Python/3.9.6 Darwin/19.6.0 source/x86_64 prompt/off
command/memorydb.update-cluster",
    "requestParameters": {
        "clusterName": "memorydb-cluster",
        "snapshotWindow": "04:00-05:00",
        "shardConfiguration": {
            "shardCount": 2
        }
    },
}

```



```

"responseElements": {
  "cluster": {
    "name": "memorydb-cluster",
    "status": "updating",
    "numberOfShards": 2,
    "availabilityMode": "MultiAZ",
    "clusterEndpoint": {
      "address": "clustercfg.memorydb-cluster.cde8da.memorydb.us-
east-1.amazonaws.com",
      "port": 6379
    },
    "nodeType": "db.r6g.large",
    "engineVersion": "6.2",
    "EnginePatchVersion": "6.2.6",
    "parameterGroupName": "default.memorydb-redis6",
    "parameterGroupStatus": "in-sync",
    "subnetGroupName": "memorydb-subnet-group",
    "tLSEnabled": true,
    "aRN": "arn:aws:memorydb:us-east-1:123456789012:cluster/memorydb-cluster",
    "snapshotRetentionLimit": 0,
    "maintenanceWindow": "tue:06:30-tue:07:30",
    "snapshotWindow": "04:00-05:00",
    "autoMinorVersionUpgrade": true,
    "DataTiering": "false"
  }
},
"requestID": "dad021ce-d161-4365-8085-574133afab54",
"eventID": "e0120f85-ab7e-4ad4-ae78-43ba15dee3d8",
"readOnly": false,
"eventType": "AwsApiCall",
"managementEvent": true,
"recipientAccountId": "123456789012",
"eventCategory": "Management"
}

```

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan CreateUser tindakan. Perhatikan bahwa untuk panggilan MemoryDB yang berisi data sensitif, data tersebut akan disunting dalam CloudTrail peristiwa yang sesuai seperti yang ditunjukkan pada bagian di bawah inirequestParameters.

```

{
  "eventVersion": "1.08",
  "userIdentity": {

```

```

    "type": "IAMUser",
    "principalId": "EKIAUAXQT3SWDEXAMPLE",
    "arn": "arn:aws:iam::123456789012:user/john",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "john"
  },
  "eventTime": "2021-07-10T19:56:13Z",
  "eventSource": "memorydb.amazonaws.com",
  "eventName": "CreateUser",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.01",
  "userAgent": "aws-cli/2.2.29 Python/3.9.6 Darwin/19.6.0 source/x86_64 prompt/off
command/memorydb.create-user",
  "requestParameters": {
    "userName": "memorydb-user",
    "authenticationMode": {
      "type": "password",
      "passwords": [
        "HIDDEN_DUE_TO_SECURITY_REASONS"
      ]
    }
  },
  "accessString": "~* &* -@all +@read"
},
"responseElements": {
  "user": {
    "name": "memorydb-user",
    "status": "active",
    "accessString": "off ~* &* -@all +@read",
    "aCLNames": [],
    "minimumEngineVersion": "6.2",
    "authentication": {
      "type": "password",
      "passwordCount": 1
    },
    "aRN": "arn:aws:memorydb:us-east-1:123456789012:user/memorydb-user"
  }
},
"requestID": "ae288b5e-80ab-4ff8-989a-5ee5c67cd193",
"eventID": "ed096e3e-16f1-4a23-866c-0baa6ec769f6",
"readOnly": false,
"eventType": "AwsApiCall",
"managementEvent": true,
"recipientAccountId": "123456789012",

```

```
"eventCategory": "Management"  
}
```

Validasi kepatuhan untuk MemoryDB

Auditor pihak ketiga menilai keamanan dan kepatuhan MemoryDB sebagai bagian dari beberapa AWS program kepatuhan. Hal ini mencakup:

- Standar Keamanan Data Industri Kartu Pembayaran (PCI DSS). Untuk informasi lain, lihat [PCI DSS](#).
- Undang-Undang Akuntabilitas dan Portabilitas Asuransi Kesehatan Perjanjian Rekan Bisnis (HIPAA BAA). Untuk informasi lebih lanjut, lihat [Kepatuhan HIPAA](#).
- Kontrol Sistem dan Organisasi (SOC) 1, 2, dan 3. Untuk informasi selengkapnya, lihat [SOC](#).
- Program Manajemen Risiko dan Otorisasi Federal (FedRAMP) Moderat. Untuk informasi lebih lanjut, lihat [FedRAMP](#).
- ISO/IEC 27001:2013, 27017:2015, 27018:2019, dan ISO/IEC 9001:2015. Untuk informasi selengkapnya, lihat [sertifikasi dan layanan AWS ISO dan CSA STAR](#).

Untuk daftar AWS layanan dalam lingkup program kepatuhan tertentu, lihat [AWS Layanan dalam Lingkup berdasarkan Program Kepatuhan](#).

Anda dapat mengunduh laporan audit pihak ketiga menggunakan AWS Artifact. Untuk informasi selengkapnya, lihat [Mengunduh Laporan di AWS Artifact](#).

Tanggung jawab kepatuhan Anda saat menggunakan MemoryDB ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, dan hukum dan peraturan yang berlaku. AWS menyediakan sumber daya berikut untuk membantu kepatuhan:

- [Panduan Quick Start Keamanan dan Kepatuhan](#) – Panduan deployment ini membahas pertimbangan arsitektur dan menyediakan langkah-langkah untuk melakukan deployment terhadap lingkungan dasar di AWS yang menjadi fokus keamanan dan kepatuhan.
- [AWS Sumber Daya AWS](#) — Kumpulan buku kerja dan panduan ini mungkin berlaku untuk industri dan lokasi Anda.
- [Mengevaluasi Sumber Daya dengan Aturan](#) dalam Panduan Developer AWS Config – AWS Config menilai seberapa patuh konfigurasi sumber daya Anda terhadap praktik internal, panduan industri, dan peraturan.

- [AWS Security Hub](#)— AWS Layanan ini memberikan pandangan komprehensif tentang keadaan keamanan Anda di dalamnya AWS yang membantu Anda memeriksa kepatuhan Anda terhadap standar industri keamanan dan praktik terbaik.
- [AWS Audit Manager](#) — AWS Layanan ini membantu Anda terus mengaudit AWS penggunaan Anda untuk menyederhanakan cara Anda mengelola risiko dan kepatuhan terhadap peraturan dan standar industri.

Keamanan infrastruktur di MemoryDB

Sebagai layanan terkelola, MemoryDB dilindungi oleh prosedur keamanan jaringan AWS global yang dijelaskan dalam whitepaper [Amazon Web Services: Overview of Security Processes](#).

Anda menggunakan panggilan API yang AWS dipublikasikan untuk mengakses MemoryDB melalui jaringan. Klien harus mendukung Keamanan Lapisan Pengangkutan (TLS) 1.2 atau versi yang lebih baru. Kami merekomendasikan TLS 1.3 atau versi yang lebih baru. Klien juga harus mendukung cipher suite dengan perfect forward secrecy (PFS) seperti Ephemeral Diffie-Hellman (DHE) atau Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Sebagian besar sistem-sistem modern seperti Java 7 dan versi yang lebih baru mendukung mode-mode ini.

Selain itu, permintaan harus ditandatangani menggunakan ID kunci akses dan kunci akses rahasia yang dikaitkan dengan pengguna utama IAM. Atau Anda dapat menggunakan [AWS Security Token Service](#) (AWS STS) untuk membuat kredensial keamanan sementara untuk menandatangani permintaan.

Privasi lalu lintas antar jaringan

MemoryDB menggunakan teknik berikut untuk mengamankan data Anda dan melindunginya dari akses yang tidak sah:

- [MemoryDB dan Amazon VPC](#) menjelaskan jenis grup keamanan yang Anda butuhkan untuk instalasi Anda.
- [MemoryDB API dan antarmuka titik akhir VPC \(\)AWS PrivateLink](#) memungkinkan Anda membuat koneksi pribadi antara titik akhir VPC dan MemoryDB API Anda.
- [Manajemen identitas dan akses di MemoryDB](#) untuk memberikan dan membatasi tindakan pengguna, grup, dan peran.

MemoryDB dan Amazon VPC

Layanan Amazon Virtual Private Cloud (Amazon VPC) mendefinisikan jaringan virtual yang mirip dengan pusat data tradisional. Anda dapat membuat konfigurasi virtual private cloud (VPC) dengan Amazon VPC, Anda dapat memilih baris alamat IP, membuat subnet, dan mengonfigurasi tabel rute, gateway jaringan, dan pengaturan keamanan. Anda juga dapat menambahkan sebuah klaster ke jaringan virtual, dan mengontrol akses ke klaster dengan menggunakan kelompok-kelompok keamanan Amazon VPC.

Bagian ini menjelaskan cara mengonfigurasi klaster MemoryDB di VPC secara manual. Informasi ini ditujukan untuk pengguna yang menginginkan pemahaman yang lebih dalam tentang cara MemoryDB dan Amazon VPC bekerja bersama.

Topik

- [Memahami MemoryDB dan VPC](#)
- [Pola Akses untuk Mengakses Cluster MemoryDB di Amazon VPC](#)
- [Membuat Virtual Private Cloud \(VPC\)](#)

Memahami MemoryDB dan VPC

MemoryDB sepenuhnya terintegrasi dengan Amazon VPC. Untuk pengguna MemoryDB, hal ini berarti sebagai berikut:

- MemoryDB selalu meluncurkan klaster Anda di VPC.
- Jika Anda baru AWS, VPC default akan dibuat untuk Anda secara otomatis.
- Jika Anda memiliki VPC default dan tidak menentukan subnet saat meluncurkan sebuah klaster, maka klaster itu akan diluncurkan ke Amazon VPC default Anda.

Untuk informasi lain, lihat [Mendeteksi Platform Anda yang Didukung dan Apakah Anda Memiliki VPC Default](#).

Dengan Amazon VPC, Anda dapat membuat jaringan virtual di AWS Cloud yang sangat menyerupai pusat data tradisional. Anda dapat mengonfigurasi VPC Anda, termasuk memilih baris alamat IP, membuat subnet, dan mengonfigurasi tabel rute, gateway jaringan, dan pengaturan keamanan.

MemoryDB mengelola peningkatan perangkat lunak, patching, deteksi kegagalan, dan pemulihan perangkat lunak.

Sekilas MemoryDB di VPC

1

VPC adalah bagian terisolasi dari AWS Cloud yang diberikan blok alamat IP sendiri.

2

Gateway internet menghubungkan VPC Anda langsung ke internet dan menyediakan akses ke lainnya AWS sumber daya seperti Amazon Simple Storage Service (Amazon S3) yang berjalan di luar VPC Anda.

3

Subnet Amazon VPC adalah segmen rentang alamat IP VPC tempat Anda dapat mengisolasi AWS sumber daya sesuai dengan keamanan dan kebutuhan operasional Anda.

4

Tabel perutean di VPC Anda mengarahkan lalu lintas jaringan antara subnet dan internet. Amazon VPC memiliki router tersirat.

5

Grup keamanan Amazon VPC mengontrol lalu lintas masuk dan keluar untuk kluster MemoryDB dan instans Amazon EC2.

6

Anda dapat meluncurkan kluster MemoryDB di subnet. Simpul memiliki alamat IP privat dari rentang alamat subnet.

7

Anda juga dapat meluncurkan instans Amazon EC2 di subnet. Setiap instans Amazon EC2 memiliki alamat IP privat dari rentang alamat subnet. Instans Amazon EC2 dapat menyambung ke simpul apa pun di subnet yang sama.

8

Agar instans Amazon EC2 di VPC Anda dapat dicapai dari internet, Anda perlu menetapkan alamat IP publik dan statis yang disebut alamat IP Elastis untuk instans tersebut.

Prasyarat

Untuk membuat kluster MemoryDB dalam VPC, VPC Anda harus memenuhi persyaratan berikut:

- VPC Anda harus mengizinkan instans Amazon EC2 nondedicated. Anda tidak dapat menggunakan MemoryDB di VPC yang mempunyai konfigurasi untuk penghuni instans khusus.
- Grup subnet harus ditentukan untuk VPC Anda. MemoryDB menggunakan grup subnet untuk memilih subnet dan alamat IP dalam subnet tersebut untuk mengasosiasikan dengan simpul Anda.
- Grup keamanan harus didefinisikan untuk VPC Anda, atau Anda dapat menggunakan grup default yang disediakan.
- Blok CIDR untuk setiap subnet harus cukup besar untuk menyediakan alamat IP cadangan untuk MemoryDB untuk digunakan selama kegiatan pemeliharaan.

Perutean dan keamanan

Anda dapat membuat konfigurasi perutean di VPC Anda untuk mengontrol aliran lalu lintas (misalnya, ke gateway internet atau gateway privat virtual). Dengan gateway internet, VPC Anda memiliki akses langsung ke yang lainAWSsumber daya yang tidak berjalan di VPC Anda. Jika Anda memilih untuk hanya memiliki gateway privat virtual dengan koneksi ke jaringan lokal dari organisasi Anda, maka Anda dapat merutekan lalu lintas dari dan ke internet tersebut melalui VPN dan menggunakan kebijakan keamanan lokal dan firewall untuk mengontrol egress. Dalam kasus ini, Anda dikenakan biaya bandwidth tambahan saat Anda mengaksesAWSsumber daya melalui internet.

Anda dapat menggunakan kelompok-kelompok keamanan Amazon VPC untuk membantu mengamankan kluster MemoryDB dan instans Amazon EC2 di Amazon VPC Anda. Grup keamanan bertindak seperti firewall di tingkat instans, bukan di tingkat subnet.

Note

Sangat dianjurkan untuk menggunakan nama DNS untuk terhubung ke simpul Anda, karena alamat IP yang mendasari dapat berubah seiring waktu.

Dokumentasi Amazon VPC

Amazon VPC memiliki kumpulan dokumentasinya sendiri untuk menjelaskan cara membuat dan menggunakan Amazon VPC Anda. Tabel berikut menunjukkan tempat menemukan informasi dalam panduan Amazon VPC.

Deskripsi	Dokumentasi
Cara memulai menggunakan Amazon VPC	Memulai dengan Amazon VPC
Cara menggunakan Amazon VPC melalui AWS Management Console	Panduan Pengguna Amazon VPC
Deskripsi lengkap dari semua perintah Amazon VPC	Referensi Baris Perintah Amazon EC2 (perintah Amazon VPC ditemukan di referensi Amazon EC2)
Deskripsi lengkap dari operasi API Amazon VPC, tipe data, dan kesalahan	Referensi API Amazon EC2 (operasi Amazon VPC API ditemukan di referensi Amazon EC2)
Informasi untuk administrator jaringan yang perlu membuat konfigurasi gateway di ujung koneksi VPN IPsec opsional Anda	ApaAWS Site-to-Site VPN?

Untuk informasi selengkapnya tentang Amazon Virtual Private Cloud, lihat [Amazon Virtual Private Cloud](#).

Pola Akses untuk Mengakses Cluster MemoryDB di Amazon VPC

MemoryDB mendukung skenario berikut untuk mengakses cluster di VPC Amazon:

Daftar Isi

- [Mengakses Cluster MemoryDB saat itu dan Instans Amazon EC2 berada di VPC Amazon yang Sama](#)
- [Mengakses Cluster MemoryDB saat itu dan Instans Amazon EC2 berada di VPC Amazon yang Berbeda](#)
 - [Mengakses Cluster MemoryDB saat itu dan Instans Amazon EC2 berada di VPC Amazon yang Berbeda di Wilayah yang Sama](#)
 - [Menggunakan Transit Gateway](#)
 - [Mengakses Cluster MemoryDB saat itu dan Instans Amazon EC2 berada di VPC Amazon yang Berbeda di Berbagai Wilayah](#)
 - [Menggunakan VPC Transit](#)
- [Mengakses Cluster MemoryDB dari Aplikasi yang Berjalan di Pusat Data Pelanggan](#)
 - [Mengakses Cluster MemoryDB dari Aplikasi yang Berjalan di Pusat Data Pelanggan Menggunakan Konektivitas VPN](#)
 - [Mengakses Cluster MemoryDB dari Aplikasi yang Berjalan di Pusat Data Pelanggan Menggunakan Direct Connect](#)

Mengakses Cluster MemoryDB saat itu dan Instans Amazon EC2 berada di VPC Amazon yang Sama

Kasus penggunaan yang paling umum adalah saat aplikasi yang di-deploy pada instans EC2 perlu terhubung ke klaster di VPC yang sama.

Cara paling sederhana untuk mengelola akses antara instans EC2 dan klaster di VPC yang sama adalah dengan melakukan tindakan berikut:

1. Buat grup keamanan VPC untuk klaster Anda. Grup keamanan ini dapat digunakan untuk membatasi akses ke cluster. Sebagai contoh, Anda dapat membuat aturan kustom untuk grup keamanan ini yang mengizinkan akses TCP menggunakan port yang Anda tetapkan untuk klaster saat Anda membuatnya dan alamat IP yang Anda gunakan untuk mengakses klaster tersebut.

Port default untuk cluster MemoryDB adalah. 6379

2. Buat grup keamanan VPC untuk instans EC2 Anda (server web dan aplikasi). Jika diperlukan, grup keamanan ini dapat mengizinkan akses ke instans EC2 dari Internet menggunakan tabel perutean VPC. Sebagai contoh, Anda dapat menetapkan aturan pada grup keamanan ini untuk mengizinkan akses TCP ke instans EC2 melalui port 22.
3. Buat aturan khusus di grup keamanan untuk klaster Anda yang memungkinkan koneksi dari grup keamanan yang Anda buat untuk instans EC2 Anda. Hal ini akan mengizinkan semua anggota grup keamanan untuk mengakses klaster.

Untuk membuat aturan dalam grup keamanan VPC yang memungkinkan koneksi dari grup keamanan lain

1. [Masuk ke Konsol AWS Manajemen dan buka konsol VPC Amazon di https://console.aws.amazon.com/vpc.](https://console.aws.amazon.com/vpc)
2. Di panel navigasi kiri, pilih Grup Keamanan.
3. Pilih atau buat grup keamanan yang akan Anda gunakan untuk cluster Anda. Pada Aturan Masuk, pilih Edit Aturan Masuk lalu pilih Tambahkan Aturan. Grup keamanan ini akan mengizinkan akses bagi anggota dari grup keamanan lain.
4. Dari Jenis, pilih Aturan TCP Kustom.
 - a. Untuk Rentang Port, tentukan port yang Anda gunakan saat membuat klaster.

Port default untuk cluster MemoryDB adalah. 6379
 - b. Pada kotak Sumber, masukkan ID dari grup keamanan. Dari daftar, pilih grup keamanan yang akan Anda gunakan untuk instans Amazon EC2 Anda.
5. Pilih Simpan jika selesai.

Mengakses Cluster MemoryDB saat itu dan Instans Amazon EC2 berada di VPC Amazon yang Berbeda

Ketika cluster Anda berada di VPC yang berbeda dari instans EC2 yang Anda gunakan untuk mengaksesnya, ada beberapa cara untuk mengakses cluster. Jika instans cluster dan EC2 berada di VPC yang berbeda tetapi di wilayah yang sama, Anda dapat menggunakan pengintip VPC. Jika cluster dan instans EC2 berada di wilayah yang berbeda, Anda dapat membuat konektivitas VPN antar wilayah.

Topik

- [Mengakses Cluster MemoryDB saat itu dan Instans Amazon EC2 berada di VPC Amazon yang Berbeda di Wilayah yang Sama](#)
- [Mengakses Cluster MemoryDB saat itu dan Instans Amazon EC2 berada di VPC Amazon yang Berbeda di Berbagai Wilayah](#)

Mengakses Cluster MemoryDB saat itu dan Instans Amazon EC2 berada di VPC Amazon yang Berbeda di Wilayah yang Sama

Cluster diakses oleh instans Amazon EC2 di VPC Amazon yang berbeda dalam Wilayah yang sama - VPC Peering Connection

Koneksi peering VPC adalah koneksi jaringan antara dua VPC yang memungkinkan Anda merutekan lalu lintas di antara keduanya menggunakan alamat IP privat. Instans di kedua VPC tersebut dapat berkomunikasi satu sama lain seolah berada di jaringan yang sama. Anda dapat membuat koneksi peering VPC antara VPC Amazon Anda sendiri, atau dengan VPC Amazon di akun lain dalam satu AWS wilayah. Untuk mempelajari selengkapnya peering Amazon VPC, lihat [Dokumentasi VPC](#).

Untuk mengakses cluster di VPC Amazon yang berbeda melalui peering

1. Pastikan bahwa kedua VPC tidak memiliki rentang IP yang tumpang-tindih atau Anda tidak akan dapat melakukan peering ke VPC tersebut.
2. Lakukan peering di antara kedua VPC. Untuk informasi selengkapnya, lihat [Membuat dan Menerima Koneksi Peering VPC Amazon](#).
3. Perbarui tabel rute Anda. Untuk informasi selengkapnya, lihat [Memperbarui Tabel Rute Anda untuk Koneksi Peering VPC](#)
4. Ubah Grup Keamanan klaster MemoryDB Anda untuk memungkinkan koneksi masuk dari grup keamanan Aplikasi di VPC peered. Untuk informasi lain, lihat [Grup Keamanan VPC Peer Referensi](#).

Mengakses klaster melalui koneksi peering akan dikenakan biaya transfer data tambahan.

Menggunakan Transit Gateway

Gateway transit memungkinkan Anda untuk melampirkan VPC dan koneksi VPN di AWS Wilayah yang sama dan merutekan lalu lintas di antara mereka. Gateway transit berfungsi di seluruh AWS akun, dan Anda dapat menggunakan AWS Resource Access Manager untuk berbagi gateway transit Anda dengan akun lain. Setelah Anda berbagi gateway transit dengan AWS akun lain, pemilik akun dapat melampirkan VPC mereka ke gateway transit Anda. Pengguna dari kedua akun ini dapat menghapus lampiran VPC tersebut kapan saja.

Anda dapat mengaktifkan multicast pada gateway transit, lalu membuat domain multicast gateway transit yang memungkinkan lalu lintas multicast dikirim dari sumber multicast Anda untuk anggota grup multicast melalui lampiran VPC yang Anda kaitkan dengan domain.

Anda juga dapat membuat lampiran koneksi peering antara gateway transit di Wilayah yang berbeda. AWS Hal ini memungkinkan Anda merutekan lalu lintas di antara beberapa lampiran gateway transit di Wilayah yang berbeda.

Untuk informasi selengkapnya, lihat [Gateway transit](#).

Mengakses Cluster MemoryDB saat itu dan Instans Amazon EC2 berada di VPC Amazon yang Berbeda di Berbagai Wilayah

Menggunakan VPC Transit

Selain menggunakan peering VPC, strategi umum lain untuk menghubungkan beberapa VPC dan jaringan jarak jauh yang tersebar secara geografis adalah membuat VPC transit yang berfungsi sebagai pusat transit jaringan global. VPC transit menyederhanakan manajemen jaringan dan meminimalkan jumlah koneksi yang diperlukan untuk menghubungkan beberapa VPC dan jaringan jarak jauh. Rancangan ini dapat menghemat waktu dan tenaga serta mengurangi biaya karena diimplementasikan secara virtual tanpa biaya yang biasa diperlukan untuk membangun kehadiran fisik di hub transit kolokasi atau men-deploy peralatan jaringan fisik.

Menghubungkan ke VPC yang berbeda-beda di wilayah yang berbeda-beda

Setelah VPC Transit Amazon dibuat, aplikasi yang digunakan dalam VPC “spoke” di satu wilayah dapat terhubung ke cluster MemoryDB di VPC “spoke” di wilayah lain.

Untuk mengakses cluster di VPC yang berbeda dalam Wilayah yang berbeda AWS

1. Lakukan deployment Solusi VPC Transit. Untuk informasi selengkapnya, lihat [AWS Transit Gateway](#).

2. Perbarui tabel perutean VPC di Aplikasi dan VPC untuk merutekan lalu lintas melalui VGW (Virtual Private Gateway) dan VPN Appliance. Dalam kasus Perutean Dinamis dengan Protokol Gateway Batas (BGP), rute Anda dapat disebarakan secara otomatis.
3. Ubah Grup Keamanan klaster MemoryDB Anda untuk memungkinkan koneksi masuk dari rentang IP instance Aplikasi. Perhatikan bahwa Anda tidak akan dapat mereferensikan Grup Keamanan server aplikasi dalam skenario ini.

Mengakses klaster di seluruh wilayah akan memperkenalkan latensi jaringan dan biaya transfer data lintas wilayah tambahan.

Mengakses Cluster MemoryDB dari Aplikasi yang Berjalan di Pusat Data Pelanggan

Skenario lain yang mungkin adalah arsitektur Hybrid di mana klien atau aplikasi di pusat data pelanggan mungkin perlu mengakses Cluster MemoryDB di VPC. Skenario ini juga didukung jika ada konektivitas antara VPC pelanggan dan pusat data baik melalui VPN atau Direct Connect.

Topik

- [Mengakses Cluster MemoryDB dari Aplikasi yang Berjalan di Pusat Data Pelanggan Menggunakan Konektivitas VPN](#)
- [Mengakses Cluster MemoryDB dari Aplikasi yang Berjalan di Pusat Data Pelanggan Menggunakan Direct Connect](#)

Mengakses Cluster MemoryDB dari Aplikasi yang Berjalan di Pusat Data Pelanggan Menggunakan Konektivitas VPN

Menghubungkan ke MemoryDB dari pusat data Anda melalui VPN

Untuk mengakses klaster di VPC dari aplikasi on-prem melalui koneksi VPN

1. Membuat Konektivitas VPN dengan menambahkan perangkat keras Gateway Privat Virtual ke VPC Anda. Untuk informasi selengkapnya, lihat [Menambahkan Gateway Privat Virtual Perangkat Keras ke VPC Anda](#).
2. Perbarui tabel perutean VPC untuk subnet tempat klaster MemoryDB Anda digunakan untuk memungkinkan lalu lintas dari server aplikasi lokal Anda. Dalam kasus Perutean Dinamis dengan BGP, rute Anda dapat disebarakan secara otomatis.

3. Ubah Grup Keamanan klaster MemoryDB Anda untuk mengizinkan koneksi masuk dari server aplikasi lokal.

Mengakses klaster melalui koneksi VPN akan menimbulkan latensi jaringan dan biaya transfer data tambahan.

Mengakses Cluster MemoryDB dari Aplikasi yang Berjalan di Pusat Data Pelanggan Menggunakan Direct Connect

Menghubungkan ke MemoryDB dari pusat data Anda melalui Direct Connect

Untuk mengakses cluster MemoryDB dari aplikasi yang berjalan di jaringan Anda menggunakan Direct Connect

1. Buat konektivitas Direct Connect. Untuk informasi selengkapnya, lihat [Memulai dengan AWS Direct Connect](#).
2. Ubah Grup Keamanan klaster MemoryDB Anda untuk mengizinkan koneksi masuk dari server aplikasi lokal.

Mengakses klaster melalui koneksi DX dapat menimbulkan latensi jaringan dan biaya transfer data tambahan.

Membuat Virtual Private Cloud (VPC)

Dalam contoh ini, Anda membuat virtual private cloud (VPC) berdasarkan layanan Amazon VPC dengan subnet privat untuk setiap Availability Zone.

Membuat VPC (Console)

Untuk membuat klaster MemoryDB di dalam Amazon Virtual Private Cloud

1. Masuk keAWSManagement Console, dan buka konsol Amazon VPC di<https://console.aws.amazon.com/vpc/>.
2. Di dasbor VPC, pilihBuat VPC.
3. Di bawahSumber dayauntuk membuat, memilihVPC dan banyak lagi.
4. Di bawahJumlah Availability Zone (AZ), pilih jumlah Availability Zone yang ingin Anda luncurkan subnet Anda.
5. Di bawahJumlah subnet, pilih jumlah subnet publik yang ingin Anda tambahkan ke VPC Anda.
6. Di bawahJumlah subnet, pilih jumlah subnet pribadi yang ingin Anda tambahkan ke VPC Anda.

Tip

Perhatikan pengidentifikasi subnet Anda, dan yang publik dan privat. Anda akan membutuhkan informasi ini nanti saat meluncurkan klaster dan menambahkan instans Amazon EC2 ke Amazon VPC Anda.

7. Buat grup keamanan Amazon VPC. Anda akan menggunakan grup ini untuk klaster dan instans Amazon EC2 Anda.
 - a. Di panel navigasi kiriAWS Management Console, pilihKelompok Keamanan.
 - b. Pilih Create Security Group (Buat Grup Keamanan).
 - c. Masukkan nama dan deskripsi untuk grup keamanan Anda di kotak yang sesuai. UntukVPC, pilih pengenalan untuk VPC Anda.
 - d. Bila pengaturan sudah sesuai keinginan Anda, pilih Ya, Buat.
8. Tentukan aturan ingress jaringan untuk grup keamanan Anda. Aturan ini akan memungkinkan Anda terhubung ke instans Amazon EC2 Anda menggunakan Secure Shell (SSH).
 - a. Di panel navigasi sebelah kiri, pilih Grup Keamanan.
 - b. Temukan grup keamanan Anda di dalam daftar, dan kemudian pilih grup itu.

- c. Di bawah Grup Keamanan, pilih tab Masuk. Pada kotak Buat aturan baru, pilih SSH, lalu pilih Tambahkan Aturan.

Tetapkan nilai berikut untuk aturan masuk baru Anda untuk memungkinkan akses HTTP:

- Jenis: HTTP
- Sumber: 0.0.0.0/0

- d. Tetapkan nilai berikut untuk aturan masuk baru Anda untuk memungkinkan akses HTTP:

- Jenis: HTTP
- Sumber: 0.0.0.0/0

Pilih Terapkan Perubahan Aturan.

Sekarang Anda siap untuk membuat [grup](#) dan [membuat sebuah klaster](#) di VPC Anda.

Subnet dan grup subnet

Grup subnet adalah kumpulan subnet (biasanya pribadi) yang dapat Anda tentukan untuk kluster yang berjalan di lingkungan Amazon Virtual Private Cloud (VPC).

Saat membuat kluster di VPC Amazon, Anda dapat menentukan grup subnet atau menggunakan grup default yang disediakan. MemoryDB menggunakan grup subnet itu untuk memilih subnet dan alamat IP dalam subnet itu untuk dikaitkan dengan node Anda.

Bagian ini mencakup cara membuat dan memanfaatkan subnet dan grup subnet untuk mengelola akses ke sumber daya MemoryDB Anda.

Untuk informasi selengkapnya tentang penggunaan grup subnet di lingkungan Amazon VPC, lihat [Langkah 2: Otorisasi akses ke cluster](#).

ID MemoryDB AZ yang didukung

Nama Wilayah/Wilayah	ID AZ yang didukung		
Wilayah AS Timur (Ohio) us-east-2	use2-az1, use2-az2, use2-az3		
Wilayah AS Timur (Virginia Utara) us-east-1	use1-az2, use1-az4, use1-az6		
Wilayah AS Barat (California Utara) us-west-1	usw1-az1, usw1-az2, usw1-az3		
Wilayah AS Barat (Oregon) us-west-2	usw2-az1, usw2-az2, usw2-az3		
Wilayah Kanada (Pusat)	cac1-az1, cac1-az2, cac1-az4		

Nama Wilayah/Wilayah	ID AZ yang didukung		
ca-central-1			
Wilayah Asia Pasifik (Hong Kong) ap-east-1	ape1-az1, ape1-az2, ape1-az3		
Wilayah Asia Pasifik (Mumbai) ap-south-1	aps1-az1, aps1-az2, aps1-az3		
Wilayah Asia Pasifik (Tokyo) ap-northeast-1	apne1-az1, apne1-az2, apne1-az4		
Wilayah Asia Pasifik (Seoul) ap-northeast-2	apne2-az1, apne2-az2, apne2-az3		
Wilayah Asia Pasifik (Singapura) ap-southeast-1	apse1-az1, apse1-az2, apse1-az3		
Wilayah Asia Pasifik (Sydney) ap-southeast-2	apse2-az1, apse2-az2, apse2-az3		
Wilayah Eropa (Frankfurt) eu-central-1	euc1-az1, euc1-az2, euc1-az3		

Nama Wilayah/Wilayah	ID AZ yang didukung		
Wilayah Eropa (Irlandia) eu-west-1	euw1-az1, euw1-az2, euw1-az3		
Wilayah Eropa (London) eu-west-2	euw2-az1, euw2-az2, euw2-az3		
Wilayah Eropa (Paris) eu-west-3	euw3-az1, euw3-az2, euw3-az3		
Wilayah Eropa (Stockholm) eu-north-1	eun1-az1, eun1-az2, eun1-az3		
Wilayah Eropa (Milan) eu-south-1	eus1-az1, eus1-az2, eus1-az3		
Wilayah Amerika Selatan (Sao Paulo) sa-east-1	sae1-az1, sae1-az2, sae1-az3		
Wilayah Tiongkok (Beijing) cn-north-1	cnn1-az1, cnn1-az2		
Wilayah China (Ningxia) cn-northwest-1	cnw1-az1, cnw1-az2, cnw1-az3		

Topik

- [Membuat grup subnet](#)
- [Memperbarui grup subnet](#)
- [Melihat detail grup subnet](#)
- [Menghapus grup subnet](#)

Membuat grup subnet

Saat Anda membuat grup subnet baru, perhatikan jumlah alamat IP yang tersedia. Jika subnet memiliki sangat sedikit alamat IP yang bebas, Anda akan dibatasi dalam hal jumlah simpul yang dapat ditambahkan ke klaster. Untuk mengatasi masalah ini, Anda dapat menetapkan satu atau beberapa subnet ke grup subnet sehingga Anda memiliki jumlah alamat IP yang cukup dalam Zona Ketersediaan dari klaster Anda. Setelah itu, Anda dapat menambahkan lebih banyak simpul ke klaster Anda.

Prosedur berikut menunjukkan kepada Anda cara membuat grup subnet yang disebut `mysubnetgroup` (konsol), API AWS CLI, dan MemoryDB.

Membuat grup subnet (Konsol)

Prosedur berikut menunjukkan cara membuat grup subnet (konsol).

Untuk membuat grup subnet (Konsol)

1. [Masuk ke AWS Management Console, dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/.](https://console.aws.amazon.com/memorydb/)
2. Di panel navigasi kiri, pilih Grup Subnet.
3. Pilih Buat Grup Subnet.
4. Di halaman Create Subnet Group, lakukan hal berikut:
 - a. Pada kotak Nama, ketik nama grup subnet Anda.

Batasan penamaan klaster adalah sebagai berikut:
 - Harus berisi 1–40 karakter alfanumerik atau tanda hubung.
 - Harus diawali dengan huruf.
 - Tidak boleh berisi dua tanda hubung berurutan.
 - Tidak boleh diakhiri dengan tanda hubung.
 - b. Di kotak Deskripsi, ketik deskripsi untuk grup subnet Anda.
 - c. Pada kotak ID VPC, pilih Amazon VPC yang Anda buat. Jika Anda belum membuatnya, pilih tombol Buat VPC dan ikuti langkah-langkah untuk membuatnya.
 - d. Di subnet yang dipilih, pilih Availability Zone dan ID subnet pribadi Anda, lalu pilih Pilih.
5. Untuk Tag, Anda dapat menerapkan tag secara opsional untuk mencari dan memfilter subnet Anda atau melacak biaya Anda AWS .

6. Jika semua pengaturan sudah sesuai keinginan Anda, pilih Buat.
7. Pada pesan konfirmasi yang muncul, pilih Tutup.

Grup subnet baru Anda muncul di daftar Grup Subnet konsol MemoryDB. Di bagian bawah jendela, Anda dapat memilih grup subnet untuk melihat detailnya, misalnya semua subnet yang terkait dengan grup ini.

Membuat grup subnet (AWS CLI)

Pada prompt perintah, gunakan perintah `create-subnet-group` untuk membuat grup subnet.

Untuk Linux, macOS, atau Unix:

```
aws memorydb create-subnet-group \  
  --subnet-group-name mysubnetgroup \  
  --description "Testing" \  
  --subnet-ids subnet-53df9c3a
```

Untuk Windows:

```
aws memorydb create-subnet-group ^  
  --subnet-group-name mysubnetgroup ^  
  --description "Testing" ^  
  --subnet-ids subnet-53df9c3a
```

Perintah ini seharusnya menghasilkan output yang serupa dengan yang berikut:

```
{  
  "SubnetGroup": {  
    "Subnets": [  
      {  
        "Identifier": "subnet-53df9c3a",  
        "AvailabilityZone": {  
          "Name": "us-east-1a"  
        }  
      }  
    ],  
    "VpcId": "vpc-3cfaef47",  
    "Name": "mysubnetgroup",  
    "ARN": "arn:aws:memorydb:us-east-1:012345678912:subnetgroup/  
mysubnetgroup",
```

```
        "Description": "Testing"  
    }  
}
```

Untuk informasi lebih lanjut, lihat AWS CLI topiknya [create-subnet-group](#).

Membuat grup subnet (MemoryDB API)

Menggunakan MemoryDB API, panggil `CreateSubnetGroup` dengan parameter berikut:

- `SubnetGroupName`=*mysubnetgroup*
- `Description`=*Testing*
- `SubnetIds.member.1`=*subnet-53df9c3a*

Memperbarui grup subnet

Anda dapat memperbarui deskripsi grup subnet, atau memodifikasi daftar ID subnet yang terkait dengan grup subnet. Anda tidak dapat menghapus ID subnet dari grup subnet jika klaster saat ini menggunakan subnet tersebut.

Prosedur berikut menunjukkan cara memperbarui grup subnet.

Memperbarui grup subnet (Konsol)

Untuk memperbarui grup subnet

1. [Masuk ke AWS Management Console dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/.](https://console.aws.amazon.com/memorydb/)
2. Di panel navigasi kiri, pilih Grup Subnet.
3. Pada daftar grup subnet, pilih grup subnet yang ingin Anda ubah.
4. Bidang nama, VPCid, dan Deskripsi tidak dapat dimodifikasi.
5. Di bagian Subnet yang dipilih klik Kelola untuk membuat perubahan apa pun pada Availability Zone yang Anda butuhkan untuk subnet. Untuk menyimpan perubahan Anda, pilih Simpan.

Memperbarui grup subnet (AWS CLI)

Pada prompt perintah, gunakan perintah `update-subnet-group` untuk memperbarui grup subnet.

Untuk Linux, macOS, atau Unix:

```
aws memorydb update-subnet-group \  
  --subnet-group-name mysubnetgroup \  
  --description "New description" \  
  --subnet-ids "subnet-42df9c3a" "subnet-48fc21a9"
```

Untuk Windows:

```
aws memorydb update-subnet-group ^  
  --subnet-group-name mysubnetgroup ^  
  --description "New description" ^  
  --subnet-ids "subnet-42df9c3a" "subnet-48fc21a9"
```

Perintah ini seharusnya menghasilkan output yang serupa dengan yang berikut:


```
{
  "SubnetGroup": {
    "VpcId": "vpc-73cd3c17",
    "Description": "New description",
    "Subnets": [
      {
        "Identifier": "subnet-42dcf93a",
        "AvailabilityZone": {
          "Name": "us-east-1a"
        }
      },
      {
        "Identifier": "subnet-48fc12a9",
        "AvailabilityZone": {
          "Name": "us-east-1a"
        }
      }
    ],
    "Name": "mysubnetgroup",
    "ARN": "arn:aws:memorydb:us-east-1:012345678912:subnetgroup/mysubnetgroup",
  }
}
```

Untuk informasi lebih lanjut, lihat AWS CLI topiknya [update-subnet-group](#).

Memperbarui grup subnet (MemoryDB API)

Menggunakan MemoryDB API, panggil UpdateSubnetGroup dengan parameter berikut:

- SubnetGroupName=*mysubnetgroup*
- Parameter lain apa pun yang nilainya ingin Anda ubah. Contoh ini menggunakan Description=*New%20description* untuk mengubah deskripsi grup subnet.

Example

```
https://memory-db.us-east-1.amazonaws.com/
?Action=UpdateSubnetGroup
&Description=New%20description
&SubnetGroupName=mysubnetgroup
&SubnetIds.member.1=subnet-42df9c3a
&SubnetIds.member.2=subnet-48fc21a9
```

```
&SignatureMethod=HmacSHA256
&SignatureVersion=4
&Timestamp=20141201T220302Z
&Version=2014-12-01
&X-Amz-Algorithm=Amazon4-HMAC-SHA256
&X-Amz-Credential=<credential>
&X-Amz-Date=20141201T220302Z
&X-Amz-Expires=20141201T220302Z
&X-Amz-Signature=<signature>
&X-Amz-SignedHeaders=Host
```

Note

Saat Anda membuat grup subnet baru, perhatikan jumlah alamat IP yang tersedia. Jika subnet memiliki sangat sedikit alamat IP yang bebas, Anda akan dibatasi dalam hal jumlah simpul yang dapat ditambahkan ke kluster. Untuk mengatasi masalah ini, Anda dapat menetapkan satu atau beberapa subnet ke grup subnet sehingga Anda memiliki jumlah alamat IP yang cukup dalam Zona Ketersediaan dari kluster Anda. Setelah itu, Anda dapat menambahkan lebih banyak simpul ke kluster Anda.

Melihat detail grup subnet

Prosedur berikut menunjukkan kepada Anda cara melihat detail grup subnet.

Melihat detail grup subnet (konsol)

Untuk melihat detail grup subnet (Konsol)

1. [Masuk ke AWS Management Console dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/.](https://console.aws.amazon.com/memorydb/)
2. Di panel navigasi kiri, pilih Grup Subnet.
3. Pada halaman grup Subnet, pilih grup subnet di bawah Nama atau masukkan nama grup subnet di bilah pencarian.
4. Pada halaman grup Subnet, pilih grup subnet di bawah Nama atau masukkan nama grup subnet di bilah pencarian.
5. Di bawah Pengaturan grup Subnet, Anda dapat melihat nama, deskripsi, ID VPC, dan Nama Sumber Daya Amazon (ARN) dari grup subnet.

6. Di bawah Subnet, Anda dapat melihat Availability Zones, Subnet ID, dan blok CIDR dari grup subnet
7. Di bawah Tag Anda dapat melihat tag apa pun yang terkait dengan grup subnet.

Melihat detail grup subnet (AWS CLI)

Pada prompt perintah, gunakan perintah `describe-subnet-groups` untuk melihat detail grup subnet tertentu.

Untuk Linux, macOS, atau Unix:

```
aws memorydb describe-subnet-groups \  
  --subnet-group-name mysubnetgroup
```

Untuk Windows:

```
aws memorydb describe-subnet-groups ^  
  --subnet-group-name mysubnetgroup
```

Perintah ini seharusnya menghasilkan output yang serupa dengan yang berikut:

```
{  
  "subnetgroups": [  
    {  
      "Subnets": [  
        {  
          "Identifier": "subnet-060cae3464095de6e",  
          "AvailabilityZone": {  
            "Name": "us-east-1a"  
          }  
        },  
        {  
          "Identifier": "subnet-049d11d4aa78700c3",  
          "AvailabilityZone": {  
            "Name": "us-east-1c"  
          }  
        },  
        {  
          "Identifier": "subnet-0389d4c4157c1edb4",  
          "AvailabilityZone": {  
            "Name": "us-east-1d"  
          }  
        }  
      ]  
    }  
  ]  
}
```

```

    }
  }
],
"VpcId": "vpc-036a8150d4300bcf2",
"Name": "mysubnetgroup",
"ARN": "arn:aws:memorydb:us-east-1:53791xzzz7620:subnetgroup/mysubnetgroup",
"Description": "test"
}
]
}

```

Untuk melihat detail pada semua grup subnet, gunakan perintah yang sama tetapi tanpa menentukan nama grup subnet.

```
aws memorydb describe-subnet-groups
```

Untuk informasi lebih lanjut, lihat AWS CLI topiknya [describe-subnet-groups](#).

Melihat grup subnet (MemoryDB API)

Menggunakan MemoryDB API, panggil DescribeSubnetGroups dengan parameter berikut:

SubnetGroupName=*mysubnetgroup*

Example

```

https://memory-db.us-east-1.amazonaws.com/
?Action=UpdateSubnetGroup
&Description=New%20description
&SubnetGroupName=mysubnetgroup
&SubnetIds.member.1=subnet-42df9c3a
&SubnetIds.member.2=subnet-48fc21a9
&SignatureMethod=HmacSHA256
&SignatureVersion=4
&Timestamp=20211801T220302Z
&Version=2021-01-01
&X-Amz-Algorithm=Amazon4-HMAC-SHA256
&X-Amz-Credential=<credential>
&X-Amz-Date=20210801T220302Z
&X-Amz-Expires=20210801T220302Z
&X-Amz-Signature=<signature>

```

```
&X-Amz-SignedHeaders=Host
```

Menghapus grup subnet

Jika Anda memutuskan bahwa Anda tidak lagi memerlukan grup subnet, Anda dapat menghapusnya. Anda tidak dapat menghapus grup subnet jika saat ini digunakan oleh sebuah klaster. Anda juga tidak dapat menghapus grup subnet pada klaster dengan pengaktifan Multi-AZ jika melakukannya meninggalkan klaster dengan kurang dari dua subnet. Anda harus terlebih dahulu menghapus centang Multi-AZ dan kemudian menghapus subnet.

Prosedur berikut menunjukkan cara menghapus grup subnet.

Menghapus grup subnet (Konsol)

Untuk menghapus grup subnet

1. [Masuk ke AWS Management Console dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/.](https://console.aws.amazon.com/memorydb/)
2. Di panel navigasi kiri, pilih Grup Subnet.
3. Dalam daftar grup subnet, pilih salah satu yang ingin Anda hapus, pilih Tindakan dan kemudian pilih Hapus.

Note

Anda tidak dapat menghapus grup subnet default atau grup yang terkait dengan cluster apa pun.

4. Layar konfirmasi Hapus Grup Subnet akan muncul.
5. Untuk menghapus grup subnet, masukkan `delete` di kotak teks konfirmasi. Untuk menyimpan grup subnet, pilih Batal.

Menghapus grup subnet (CLI AWS)

Dengan menggunakan AWS CLI, panggil perintah `delete-subnet-group` dengan parameter berikut:

- `--subnet-group-name`*mysubnetgroup*

Untuk Linux, macOS, atau Unix:

```
aws memorydb delete-subnet-group \
```

```
--subnet-group-name mysubnetgroup
```

Untuk Windows:

```
aws memorydb delete-subnet-group ^  
  --subnet-group-name mysubnetgroup
```

Untuk informasi lebih lanjut, lihat AWS CLI topiknya [delete-subnet-group](#).

Menghapus grup subnet (MemoryDB API)

Menggunakan MemoryDB API, panggil DeleteSubnetGroup dengan parameter berikut:

- SubnetGroupName=*mysubnetgroup*

Example

```
https://memory-db.us-east-1.amazonaws.com/  
?Action=DeleteSubnetGroup  
&SubnetGroupName=mysubnetgroup  
&SignatureMethod=HmacSHA256  
&SignatureVersion=4  
&Timestamp=20210801T220302Z  
&Version=2021-01-01  
&X-Amz-Algorithm=Amazon4-HMAC-SHA256  
&X-Amz-Credential=<credential>  
&X-Amz-Date=20210801T220302Z  
&X-Amz-Expires=20210801T220302Z  
&X-Amz-Signature=<signature>  
&X-Amz-SignedHeaders=Host
```

Perintah ini tidak menghasilkan output.

Untuk informasi selengkapnya, lihat topik API MemoryDB. [DeleteSubnetGroup](#)

MemoryDB API dan antarmuka titik akhir VPC (AWS PrivateLink)

Anda dapat membuat koneksi pribadi antara titik akhir VPC dan Amazon MemoryDB API dengan membuat titik akhir VPC antarmuka. Endpoint antarmuka didukung oleh [AWS PrivateLink](#). AWS PrivateLink memungkinkan Anda mengakses operasi API MemoryDB secara pribadi tanpa gateway internet, perangkat NAT, koneksi VPN, atau koneksi Direct AWS Connect.

Instans di VPC Anda tidak memerlukan alamat IP publik untuk berkomunikasi dengan titik akhir API MemoryDB. Instans Anda juga tidak memerlukan alamat IP publik untuk menggunakan salah satu operasi API MemoryDB yang tersedia. Lalu lintas antara VPC dan MemoryDB Anda tidak meninggalkan jaringan Amazon. Setiap titik akhir antarmuka direpresentasikan oleh satu atau beberapa antarmuka jaringan elastis di subnet Anda. Untuk informasi selengkapnya tentang antarmuka jaringan elastis, lihat [Antarmuka jaringan elastis](#) dalam Panduan Pengguna Amazon EC2.

- Untuk informasi selengkapnya tentang titik akhir VPC, lihat Titik akhir [VPC Antarmuka \(\) di AWS PrivateLink Panduan](#) Pengguna Amazon VPC.
- Untuk informasi selengkapnya tentang operasi API MemoryDB, lihat operasi API [MemoryDB](#).

Setelah Anda membuat titik akhir VPC antarmuka, jika Anda mengaktifkan nama host [DNS pribadi untuk titik akhir, titik akhir MemoryDB](#) default (<https://memorydb.Region.amazonaws.com>) menyelesaikan ke titik akhir VPC Anda. Jika Anda tidak mengaktifkan nama host DNS privat, Amazon VPC menyediakan nama titik akhir DNS yang dapat Anda gunakan dalam format berikut:

```
VPC_Endpoint_ID.memorydb.Region.vpce.amazonaws.com
```

Untuk informasi selengkapnya, lihat [Titik Akhir VPC Antarmuka \(AWS PrivateLink\) di Panduan Pengguna Amazon VPC](#). MemoryDB mendukung panggilan ke semua [Tindakan API](#) di dalam VPC Anda.

Note

Nama host DNS privat dapat diaktifkan hanya untuk satu titik akhir VPC di VPC. Jika Anda ingin membuat titik akhir VPC tambahan maka nama host DNS privat harus dinonaktifkan.

Pertimbangan untuk titik akhir VPC

Sebelum menyiapkan titik akhir VPC antarmuka untuk titik akhir API MemoryDB, pastikan Anda meninjau [properti dan batasan titik akhir Antarmuka di](#) Panduan Pengguna Amazon VPC. Semua operasi API MemoryDB yang relevan untuk mengelola sumber daya MemoryDB tersedia dari VPC Anda menggunakan. AWS PrivateLink Kebijakan titik akhir VPC didukung untuk titik akhir API MemoryDB. Secara default, akses penuh ke operasi API MemoryDB diizinkan melalui titik akhir. Untuk informasi selengkapnya, lihat [Mengontrol Akses ke Layanan dengan titik akhir VPC](#) dalam Panduan Pengguna Amazon VPC.

Membuat titik akhir VPC antarmuka untuk API MemoryDB

Anda dapat membuat titik akhir VPC untuk API MemoryDB menggunakan konsol VPC Amazon atau AWS CLI. Untuk informasi selengkapnya, lihat [Membuat titik akhir antarmuka](#) dalam Panduan Pengguna Amazon VPC.

Setelah Anda membuat antarmuka VPC endpoint, Anda dapat mengaktifkan nama host DNS pribadi untuk titik akhir. Ketika Anda melakukannya, titik akhir MemoryDB default (<https://memorydb.REGION.amazonaws.com>) menyelesaikan ke titik akhir VPC Anda. Untuk informasi selengkapnya, lihat [Mengakses layanan melalui titik akhir antarmuka](#) dalam Panduan Pengguna Amazon VPC.

Membuat kebijakan titik akhir VPC untuk Amazon MemoryDB API

Anda dapat melampirkan kebijakan titik akhir ke titik akhir VPC Anda yang mengontrol akses ke API MemoryDB. Kebijakan menentukan informasi berikut:

- Prinsipal yang dapat melakukan tindakan.
- Tindakan yang dapat dilakukan.
- Sumber daya yang menjadi target tindakan.

Untuk informasi selengkapnya, lihat [Mengontrol Akses ke Layanan dengan titik akhir VPC](#) dalam Panduan Pengguna Amazon VPC.

Example Kebijakan titik akhir VPC untuk tindakan API MemoryDB

Berikut ini adalah contoh kebijakan endpoint untuk API MemoryDB. Saat dilampirkan ke titik akhir, kebijakan ini memberikan akses ke tindakan API MemoryDB yang terdaftar untuk semua prinsipal di semua sumber daya.

```
{
  "Statement": [{
    "Principal": "*",
    "Effect": "Allow",
    "Action": [
      "memorydb:CreateCluster",
      "memorydb:UpdateCluster",
      "memorydb:CreateSnapshot"
    ],
    "Resource": "*"
  }]
}
```

```
}
```

Example Kebijakan titik akhir VPC yang menolak semua akses dari akun tertentu AWS

Kebijakan titik akhir VPC berikut menyangkal AWS akun **123456789012** semua akses ke sumber daya menggunakan titik akhir. Kebijakan ini mengizinkan semua tindakan dari akun lainnya.

```
{
  "Statement": [{
    "Action": "*",
    "Effect": "Allow",
    "Resource": "*",
    "Principal": "*"
  },
  {
    "Action": "*",
    "Effect": "Deny",
    "Resource": "*",
    "Principal": {
      "AWS": [
        "123456789012"
      ]
    }
  }
]
}
```

Pembaruan layanan di MemoryDB

MemoryDB secara otomatis memantau armada cluster dan node Anda untuk menerapkan pembaruan layanan saat tersedia. Biasanya, Anda mengatur jendela pemeliharaan yang telah ditentukan sehingga MemoryDB dapat menerapkan pembaruan ini. Namun, dalam beberapa kasus Anda mungkin menganggap cara ini terlalu kaku dan cenderung menghambat alur bisnis Anda.

Dengan pembaruan layanan, Anda mengontrol kapan dan pembaruan mana yang diterapkan. Anda juga dapat memantau kemajuan pembaruan ini ke cluster MemoryDB pilihan Anda secara real time.

Mengelola pembaruan layanan

Pembaruan layanan MemoryDB dirilis secara teratur. Jika Anda memiliki satu atau beberapa klaster yang memenuhi syarat untuk pembaruan layanan tersebut, Anda menerima pemberitahuan melalui

email, SNS, Personal Health Dashboard (PHD), dan CloudWatch acara Amazon saat pembaruan dirilis. Pembaruan juga ditampilkan di halaman Pembaruan Layanan di konsol MemoryDB. Dengan menggunakan dasbor ini, Anda dapat melihat semua pembaruan layanan dan statusnya untuk armada MemoryDB Anda.

Anda mengontrol waktu penerapan pembaruan sebelum pembaruan otomatis dimulai. Kami sangat menyarankan agar Anda menerapkan pembaruan jenis pembaruan keamanan sesegera mungkin untuk memastikan bahwa MemoryDB Anda selalu up-to-date dengan patch keamanan saat ini.

Bagian berikut membahas opsi-opsi tersebut secara terperinci.

Topik

- [Menerapkan pembaruan layanan](#)

Menerapkan pembaruan layanan

Anda dapat memulai menerapkan pembaruan layanan untuk armada Anda sejak pembaruan berstatus tersedia. Pembaruan layanan bersifat kumulatif. Dengan kata lain, pembaruan apa pun yang belum diterapkan akan disertakan dalam pembaruan terbaru Anda.

Jika pembaruan layanan memiliki opsi pembaruan otomatis yang aktif, Anda dapat memilih untuk tidak mengambil tindakan apa pun saat pembaruan tersedia. MemoryDB akan menjadwalkan untuk menerapkan pembaruan selama jendela pemeliharaan cluster Anda setelah tanggal mulai pembaruan Otomatis. Anda akan menerima notifikasi terkait untuk setiap tahap pembaruan.

Note

Anda dapat menerapkan hanya pembaruan layanan yang berstatus tersedia atau terjadwal.

Untuk informasi selengkapnya tentang meninjau dan menerapkan pembaruan khusus layanan apa pun ke kluster MemoryDB yang berlaku, lihat. [Menerapkan pembaruan layanan menggunakan konsol](#)

Ketika pembaruan layanan baru tersedia untuk satu atau beberapa cluster MemoryDB Anda, Anda dapat menggunakan konsol MemoryDB, API, atau AWS CLI untuk menerapkan pembaruan. Bagian berikut menjelaskan opsi yang dapat Anda gunakan untuk menerapkan pembaruan.

Menerapkan pembaruan layanan menggunakan konsol

Untuk melihat daftar pembaruan layanan yang tersedia, bersama informasi lainnya, buka halaman Pembaruan Layanan pada konsol.

1. [Masuk ke AWS Management Console dan buka konsol MemoryDB di https://console.aws.amazon.com/memorydb/.](https://console.aws.amazon.com/memorydb/)
2. Pada panel navigasi, pilih Pembaruan Layanan.

Di bawah Detail pembaruan Layanan, Anda dapat melihat hal berikut:

- Nama pembaruan layanan: Nama unik pembaruan layanan
- Perbarui deskripsi: Informasi terperinci tentang pembaruan layanan
- Tanggal mulai pembaruan otomatis: Jika atribut ini disetel, MemoryDB akan mulai menjadwalkan cluster Anda untuk diperbarui secara otomatis di jendela pemeliharaan yang sesuai setelah tanggal ini. Anda akan menerima pemberitahuan sebelumnya pada jendela pemeliharaan terjadwal yang tepat, yang mungkin tidak langsung setelah tanggal mulai pembaruan Otomatis. Anda masih dapat menerapkan pembaruan ke cluster Anda kapan saja Anda memilih. Jika atribut tidak disetel, pembaruan layanan tidak diaktifkan pembaruan otomatis dan MemoryDB tidak akan memperbarui cluster Anda secara otomatis.

Di bagian Status pembaruan klaster, Anda dapat melihat daftar klaster yang berisi pembaruan yang belum diterapkan atau baru saja diterapkan. Untuk setiap klaster, Anda dapat melihat hal berikut:

- Nama klaster: Nama dari klaster
- Simpul diperbarui: Rasio simpul individual dalam klaster tertentu yang telah diperbarui atau tetap tersedia setelah pembaruan layanan tertentu.
- Jenis Pembaruan: Jenis pembaruan layanan, yaitu pembaruan keamanan atau pembaruan mesin
- Status: Status pembaruan layanan pada klaster, yang merupakan salah satu dari berikut ini:
 - tersedia: Pembaruan ini tersedia untuk klaster yang diperlukan.
 - in-progres: Pembaruan sedang diterapkan ke cluster ini.
 - dijadwalkan: Tanggal pembaruan telah dijadwalkan.
 - selesai: Pembaruan telah berhasil diterapkan. Klaster dengan status selesai akan ditampilkan selama 7 hari setelah selesai.

Jika Anda memilih salah satu atau semua klaster dengan status tersedia atau dijadwalkan, lalu memilih Terapkan sekarang, pembaruan akan mulai diterapkan pada klaster tersebut.

Menerapkan pembaruan layanan menggunakan AWS CLI

Setelah Anda menerima notifikasi bahwa pembaruan layanan telah tersedia, Anda dapat memeriksa dan menerapkannya menggunakan AWS CLI:

- Untuk mendapatkan deskripsi pembaruan layanan yang tersedia, jalankan perintah berikut:

```
aws memorydb describe-service-updates --status available
```

Untuk informasi lebih lanjut, lihat [describe-service-updates](#).

- Untuk menerapkan pembaruan layanan pada daftar klaster, jalankan perintah berikut:

```
aws memorydb batch-update-cluster --service-update  
ServiceUpdateNameToApply=sample-service-update --cluster-names cluster-1  
cluster2
```

Untuk informasi lebih lanjut, lihat [batch-update-cluster](#).

Referensi

Topik di bagian ini mencakup bekerja dengan MemoryDB API dan bagian MemoryDB dari AWS CLI. Juga disertakan pada bagian ini adalah pesan kesalahan dan notifikasi layanan yang umum.

- [Menggunakan API MemoryDB](#)
- [Referensi API MemoryDB](#)
- [MemoryDB bagian AWS CLI Referensi](#)

Menggunakan API MemoryDB

Bagian ini memberikan deskripsi berorientasi tugas tentang cara menggunakan dan menerapkan operasi MemoryDB. Untuk penjelasan lengkap tentang operasi ini, lihat [Referensi API MemoryDB](#).

Topik

- [Menggunakan API kueri](#)
- [Pustaka yang tersedia](#)
- [Memecahkan masalah aplikasi](#)

Menggunakan API kueri

Parameter kueri

Permintaan berbasis Kueri HTTP adalah permintaan HTTP yang menggunakan HTTP verb GET atau POST dan parameter Query yang bernama `Action`.

Setiap permintaan Kueri harus menyertakan beberapa parameter umum untuk menangani autentikasi dan pemilihan tindakan.

Beberapa operasi mengambil daftar parameter. Daftar ini ditentukan menggunakan notasi `param.n`. Nilai `n` adalah bilangan bulat mulai dari 1.

Autentikasi permintaan Query

Anda hanya dapat mengirim permintaan Kueri melalui HTTPS, dan Anda harus menyertakan tanda tangan di setiap permintaan Kueri. Bagian ini menjelaskan cara membuat tanda tangan. Metode yang dijelaskan dalam prosedur berikut ini dikenal sebagai tanda tangan versi 4.

Berikut ini adalah langkah dasar yang digunakan untuk memberikan autentikasi kepada AWS. Ini mengasumsikan Anda terdaftar dengan AWS dan memiliki Access Key ID dan Secret Access Key.

Proses Autentikasi Query

1. Pengirim membangun permintaan ke AWS.
2. Pengirim menghitung tanda tangan permintaan, Keyed-Hashing untuk Kode Autentikasi Pesan Berbasis Hash (HMAC) dengan fungsi hash SHA-1, seperti yang didefinisikan dalam bagian berikutnya dari topik ini.

3. Pengirim permintaan mengirimkan data permintaan, tanda tangan, dan Access Key ID (kunci-identifier dari Secret Access Key yang digunakan) untuk AWS.
4. AWS menggunakan Access Key ID untuk mencari Secret Access Key.
5. AWS menghasilkan tanda tangan dari data permintaan dan Secret Access Key menggunakan algoritma yang sama dengan yang digunakan untuk menghitung tanda tangan pada permintaan.
6. Jika tanda tangan cocok, maka permintaan tersebut dianggap otentik. Jika perbandingan gagal, maka permintaan dibuang, dan AWS menghasilkan respons kesalahan.

Note

Jika permintaan berisi parameter `Timestamp`, tanda tangan yang dihitung untuk permintaan akan kedaluwarsa dalam 15 menit setelah nilainya.

Jika permintaan berisi parameter `Expires`, tanda tangan berakhir pada waktu yang ditentukan oleh parameter `Expires`.

Untuk menghitung tanda tangan permintaan

1. Buat string kueri canonicalized yang Anda butuhkan nanti dalam prosedur ini:
 - a. Urutkan komponen string kueri UTF-8 berdasarkan nama parameter dengan pengurutan byte alami. Parameter dapat berasal dari GET URI atau dari tubuh POST (jika Content-Type adalah aplikasi/x-www-form-urlencoded).
 - b. URL mengkodekan nama parameter dan nilainya sesuai dengan aturan berikut:
 - i. Jangan melakukan encode URL karakter unreserved apa pun yang didefinisikan RFC 3986. Karakter yang tidak dicadangkan adalah A-Z, a-z, 0-9, tanda hubung (-), garis bawah (_), titik (.), dan tilde (~).
 - ii. Gunakan encode persen pada semua karakter lain dengan `%XY`, di mana X dan Y adalah karakter hex 0-9 dan huruf besar A-F.
 - iii. Gunakan encode persen pada karakter UTF-8 yang diperluas dalam bentuk `%XY%ZA...`
 - iv. Gunakan encode persen pada karakter spasi sebagai `%20` (dan bukan `+`, seperti skema pengkodean yang umum dilakukan).
 - c. Pisahkan nama parameter yang dienkodekan dari nilai yang dienkodekan dengan tanda sama dengan (=) (karakter ASCII 61), meskipun jika nilai parameter itu kosong.

- d. Pisahkan pasangan nama-nilai dengan tanda ampersand (&) (kode ASCII 38).
2. Buat string untuk menandatangani sesuai dengan pseudo-tata bahasa berikut (tanda “\n” merupakan baris baru ASCII).

```
StringToSign = HTTPVerb + "\n" +
ValueOfHostHeaderInLowercase + "\n" +
HTTPRequestURI + "\n" +
CanonicalizedQueryString <from the preceding step>
```

Komponen HTTPRequestURI adalah komponen jalur absolut HTTP dari URI hingga, tapi tidak termasuk, string query. Jika HTTPRequestURI kosong, gunakan garis miring ke depan (/).

3. Hitung HMAC sesuai RFC 2104 dengan string yang baru saja Anda buat, Secret Access Key Anda sebagai kunci, dan SHA256 atau SHA1 sebagai algoritma hash.

Untuk informasi lain, lihat <https://www.ietf.org/rfc/rfc2104.txt>.

4. Mengonversi nilai yang dihasilkan ke base64.
5. Sertakan nilai sebagai nilai dari parameter Signature dalam permintaan.

Misalnya, berikut ini adalah permintaan sampel (baris baru ditambahkan untuk kejelasan).

```
https://memory-db.us-east-1.amazonaws.com/
?Action=DescribeClusters
&ClusterName=myCluster
&SignatureMethod=HmacSHA256
&SignatureVersion=4
&Version=2021-01-01
```

Untuk string query sebelumnya, Anda akan menghitung tanda tangan HMAC atas string berikut.

```
GET\n
memory-db.amazonaws.com\n
Action=DescribeClusters
&ClusterName=myCluster
&SignatureMethod=HmacSHA256
&SignatureVersion=4
&Version=2021-01-01
&X-Amz-Algorithm=Amazon4-HMAC-SHA256
```

```
&X-Amz-Credential=AKIADQKE4SARGYLE%2F20140523%2Fus-east-1%2Fmemorydb%2Faws4_request
&X-Amz-Date=20210801T223649Z
&X-Amz-SignedHeaders=content-type%3Bhost%3Buser-agent%3Bx-amz-content-sha256%3Bx-
amz-date
    content-type:
    host:memory-db.us-east-1.amazonaws.com
    user-agent:ServicesAPICommand_Client
x-amz-content-sha256:
x-amz-date:
```

Hasilnya adalah permintaan yang ditandatangani berikut.

```
https://memory-db.us-east-1.amazonaws.com/
?Action=DescribeClusters
&ClusterName=myCluster
&SignatureMethod=HmacSHA256
&SignatureVersion=4
&Version=2021-01-01
&X-Amz-Algorithm=Amazon4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20141201/us-east-1/memorydb/aws4_request
&X-Amz-Date=20210801T223649Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=2877960fced9040b41b4feaca835fd5cfeb9264f768e6a0236c9143f915ffa56
```

Untuk informasi rinci tentang proses penandatanganan dan menghitung tanda tangan permintaan, lihat topik [Proses tanda tangan Signature Version 4](#) dan subtopiknya.

Pustaka yang tersedia

AWS menyediakan kit pengembangan perangkat lunak (SDK) untuk developer perangkat lunak yang lebih suka membangun aplikasi menggunakan API bahasa tertentu alih-alih API Kueri. SDK ini menyediakan fungsi dasar (tidak termasuk dalam API), seperti autentikasi permintaan, percobaan ulang permintaan, dan penanganan kesalahan sehingga lebih mudah untuk memulai. SDK dan sumber daya tambahan tersedia dalam bahasa pemrograman berikut:

- [Java](#)
- [Windows dan .NET](#)
- [PHP](#)
- [Python](#)

- [Ruby](#)

Untuk informasi tentang bahasa lainnya, lihat [Contoh kode & perpustakaan](#).

Memecahkan masalah aplikasi

MemoryDB memberikan kesalahan deskriptif dan spesifik untuk membantu Anda memecahkan masalah saat berinteraksi dengan API MemoryDB.

Menarik kesalahan

Biasanya, Anda ingin aplikasi Anda memeriksa apakah permintaan menimbulkan kesalahan sebelum Anda menghabiskan waktu untuk memproses hasil. Cara termudah untuk mengetahui jika terjadi kesalahan adalah dengan mencari `ERROR` node dalam respon dari MemoryDB API.

Sintaks XPath menyediakan cara sederhana untuk mencari keberadaan simpul `ERROR`, serta cara mudah untuk menarik kode kesalahan dan pesan. Snippet kode berikut menggunakan Perl dan modul `XML::XPath` untuk menentukan jika kesalahan terjadi selama permintaan. Jika terjadi kesalahan, kode akan mencetak kode kesalahan pertama dan pesan dalam tanggapannya.

```
use XML::XPath;
my $xp = XML::XPath->new(xml =>$response);
if ( $xp->find("//Error") )
{print "There was an error processing your request:\n", " Error code: ",
 $xp->findvalue("//Error[1]/Code"), "\n", " ",
 $xp->findvalue("//Error[1]/Message"), "\n\n"; }
```

Tips penyelesaian masalah

Kami merekomendasikan proses berikut untuk mendiagnosis dan menyelesaikan masalah dengan API MemoryDB.

- Verifikasi bahwa MemoryDB berjalan dengan benar.

Untuk melakukan ini, cukup buka jendela browser dan kirimkan permintaan ke layanan MemoryDB (seperti <https://memory-db.us-east-1.amazonaws.com>). Sebuah `MissingAuthenticationTokenException` atau `UnknowNOperationException` mengkonfirmasi bahwa layanan tersedia dan menanggapi permintaan.

- Periksa struktur permintaan Anda.

Setiap operasi MemoryDB memiliki halaman referensi diReferensi API MemoryDB. Periksa ulang bahwa Anda menggunakan parameter dengan benar. Untuk memberikan Anda gambaran apa yang mungkin salah, lihat contoh permintaan atau skenario pengguna untuk melihat apakah contoh itu melakukan operasi yang serupa.

- Periksa ke forum.

MemoryDB memiliki forum diskusi tempat Anda dapat mencari solusi untuk masalah yang dialami orang lain selama ini. Untuk melihat forum, kunjungi

<https://forums.aws.amazon.com/>.

Kuota untuk MemoryDB

AWS Akun Anda memiliki kuota default, sebelumnya disebut sebagai batas, untuk setiap layanan. AWS Kecuali dinyatakan lain, setiap kuota bersifat khusus per Wilayah. Anda dapat meminta peningkatan untuk beberapa kuota dan kuota lainnya tidak dapat ditingkatkan.

Untuk meminta peningkatan kuota, lihat [Meminta Peningkatan Kuota](#) dalam Panduan Pengguna Service Quotas. Jika kuota belum tersedia dalam Service Quotas, gunakan [formulir penambahan batas](#).

AWS Akun Anda memiliki kuota berikut yang terkait dengan MemoryDB.

Sumber Daya	Default
Simpul per Wilayah	300
Node per cluster per jenis instance	90
Node per pecahan	6
Grup parameter per Wilayah	150
Grup subnet per Wilayah	150
Subnet per grup subnet	20
Pengguna per grup pengguna	100
Jumlah total pengguna	1000
Jumlah grup pengguna	100

Riwayat dokumen untuk Panduan Pengguna MemoryDB

Tabel berikut menjelaskan rilis dokumentasi untuk MemoryDB.

Perubahan	Deskripsi	Tanggal
MemoryDB sekarang mendukung otentikasi pengguna menggunakan IAM	Otentikasi IAM memungkinkan Anda untuk mengautentikasi koneksi ke MemoryDB menggunakan identitas. AWS Identity and Access Management Hal ini memungkinkan Anda memperkuat model keamanan Anda dan menyederhanakan banyak tugas keamanan administratif. Untuk informasi selengkapnya, lihat Autentikasi dengan IAM .	10 Mei 2023
MemoryDB sekarang mendukung Redis OSS 7	Rilis ini membawa beberapa fitur baru ke MemoryDB: fungsi Redis OSS, perbaikan ACL, Sharded Pub/Sub dan multiplexing I/O yang disempurnakan. Untuk informasi selengkapnya, lihat versi mesin Redis OSS .	9 Mei 2023
MemoryDB sekarang menawarkan node yang dipesan	Node cadangan memberi Anda diskon yang signifikan dibandingkan dengan harga node sesuai permintaan. Node yang dicadangkan bukanlah node fisik, melainkan diskon penagihan yang diterapkan pada penggunaan node sesuai	Desember 27, 2022

permintaan di akun Anda. Untuk informasi selengkapnya, lihat [MemoryDB reserved nodes](#).

[MemoryDB sekarang mendukung Data Tiering](#)

Tingkat data MemoryDB. Anda dapat menggunakan tingkatan data sebagai cara berbiaya lebih rendah untuk menskalakan klaster Anda hingga ratusan terabyte kapasitas. Untuk informasi selengkapnya, lihat [Tingkatan data](#).

November 3, 2022

[MemoryDB sekarang mendukung format asli JavaScript Object Notation \(JSON\)](#)

Format asli JavaScript Object Notation (JSON) adalah cara sederhana dan tanpa skema untuk menyandikan kumpulan data kompleks di dalam cluster Redis OSS. Anda dapat menyimpan dan mengakses data secara native menggunakan format JavaScript Object Notation (JSON) di dalam cluster Redis OSS dan memperbarui data JSON yang disimpan dalam cluster tersebut, tanpa perlu mengelola kode khusus untuk membuat serial dan deserialisasinya. Untuk informasi selengkapnya, lihat [Memulai dengan JSON](#).

25 Mei 2022

[MemoryDB sekarang mendukung AWS PrivateLink](#)

AWS PrivateLink memungkinkan Anda mengakses operasi API MemoryDB secara pribadi tanpa gateway internet, perangkat NAT, koneksi VPN, atau koneksi Direct AWS Connect. Untuk informasi selengkapnya, lihat [MemoryDB API dan antarmuka VPC endpoint](#) ().AWS PrivateLink

24 Januari 2022

[Rilis awal](#)

Rilis awal Panduan Pengguna MemoryDB. Untuk informasi lebih lanjut, lihat [Apa itu MemoryDB?](#)

19 Agustus 2021

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