Managed databases su AWS: scegliere il servizio perfetto per soddisfare requisiti di Alta Disponibilità, Disaster Recovery e Scalabilità

29 Gennaio 2025 - 10 min. read

Amazon RDS | Aurora Serverless | Database | Disaster Recovery (DR) | High Availability (HA)

"Quando non hai nulla, non hai nulla da perdere"

Bob Dylan, Like a Rolling Stone

Se ci pensiamo bene, questa citazione ha un fondo di verità; abbiamo sempre qualcosa su cui contare: le nostre abilità, la nostra esperienza e, ovviamente, i dati. Nessuno può portarci via le nostre competenze e la nostra esperienza, ma i dati possono sparire in tanti modi e trovare la soluzione giusta per minimizzare la marea di problemi legati alla persistenza, alla disponibilità e alla scalabilità dell'archiviazione dei dati può rivelarsi un vero rompicapo.

Come sappiamo, AWS offre diversi tipi e varianti di servizi gestiti per i database; non è per niente facile scegliere quello giusto che soddisfi le nostre esigenze senza aumentare i costi in bolletta con funzionalità non necessarie.

Oggi faremo luce sulle differenze nelle declinazioni dei servizi database gestiti nel Cloud Amazon, offrendo una panoramica utile per decidere come e quando scegliere un servizio.

Engine: Amazon RDS vs Amazon Aurora

Il primo fattore di confusione quando ci si approccia ai servizi di database gestiti su AWS è la differenza tra Amazon RDS (per MsSql, MySQL, PostgreSQL..., ecc.) e Amazon Aurora (MySQL o PostgreSQL).

Amazon RDS per... (scegli il tuo engine preferito)

Quando scegliamo di avviare un'istanza RDS, dietro le quinte AWS farà il deploy e configurerà un'istanza compute con l'engine scelto e la versione supportata corrispondente.

È possibile selezionare la capacità di calcolo e lo storage, che può scalare automaticamente fino a raggiungere il limite massimo scelto. Allo stesso modo, possiamo scegliere di applicare in modo automatico le patch del sistema operativo e dell'engine (ma solo le minori) durante una finestra di manutenzione personalizzabile.

Per quanto riguarda gli aggiornamenti delle versioni principali dell'engine, è necessario invece pianificarne l'applicazione. Attenzione: nel caso gli upgrade siano rimandati per un lungo tempo, rimarremo indietro rispetto alla matrice di versioni supportate ed entreremo nella fase di "supporto esteso", con costi aggiuntivi.

Avvertenze

Ecco un esempio pratico: se avessimo scelto di fare il deploy di un'istanza Amazon RDS per MySQL nell'ottobre 2022, avremmo ricevuto tutti gli aggiornamenti minori, se non avessimo applicato gli aggiornamenti delle versioni principali saremmo entrati nel supporto esteso il 29 febbraio 2024.

Un altro punto di attenzione riguarda lo scaling dello storage: se viene raggiunto il limite superiore per lo storage allocato (anche con l'autoscaling), dovremo ridimensionarlo, provocando downtime. Inoltre, non è possibile scalare facilmente verso il basso lo storage: è infatti necessario un deployment blue/green per questo tipo di operazione.

Amazon RDS Custom

RDS Custom permette di avere accesso al sistema operativo. Viene utilizzato solitamente quando si ha a che fare con applicazioni legacy e database engine Microsoft SQL Server o Oracle. In questo modo, si modifica lo shared responsibility model tipico di RDS, dove la manutenzione del sistema operativo sottostante è a carico di AWS. Non tratteremo in dettaglio la sua architettura, ma è un'opzione per questo scenario specifico. Come sempre, raccomandiamo l'uso di servizi gestiti per ridurre lo sforzo operativo e concentrarci sul business. A questo indirizzo sono disponibili più informazioni sul modello di responsabilità condivisa per Amazon RDS Custom.

Amazon Aurora

Amazon Aurora è un servizio gestito che separa l'engine del database dallo storage. In questo modo possiamo preoccuparci meno di aspetti come la replica dello storage e il suo scaling. Come per RDS "tradizionale", dobbiamo scegliere la dimensione della parte di compute, con un modello pay-as-you-go.

Al deploy di un'istanza Amazon Aurora RDS non viene richiesta la dimensione dello storage, questo perché l'autoscaling dello spazio è garantito da funzionalità aggiuntive. Lo storage viene anche replicato automaticamente su tre Availability Zone, con due copie per ogni zona, tollerando quindi la perdita di fino a due copie dei dati.

In caso di fallimento di un'istanza, Amazon Aurora RDS tenterà di auto-ripararsi riavviando i servizi rilevanti.

Avvertenze

Dobbiamo tenere presente che una singola istanza tenterà sì di auto-ripararsi, ma si verificherà del downtime. Inoltre, le versioni dell'engine Aurora sono leggermente diverse e non seguono il ciclo di rilascio standard.

Nel caso in cui sia necessario utilizzare un insieme di funzionalità particolari (come MyISAM per MySQL), occorre verificare che siano supportate.

Aurora Serverless

Aurora serverless spinge ulteriormente il disaccoppiamento fra storage e compute, containerizzando la parte computazionale per facilitare lo scaling. Facendo il deploy di Aurora in modalità serverless, invece di scegliere la dimensione CPU e Memoria, dobbiamo selezionare la dimensione minima e massima in termini di Aurora Capacity Units (ACU).

Una ACU misura le risorse in termini di RAM, CPU e Networking; ogni ACU corrisponde a 2 GB di RAM e un valore proporzionale di CPU. Lavorare con le ACU permette di scalare "al volo" la potenza computazionale assegnata al database.

Possiamo specificare i valori minimi e massimi (da 0,5 a 256 ACU) ed anche mettere automaticamente in pausa il database, in modo da non avere addebiti durante i periodi di inattività (come nel caso di un cluster di sviluppo o a un workload utilizzato solo durante l'orario lavorativo).

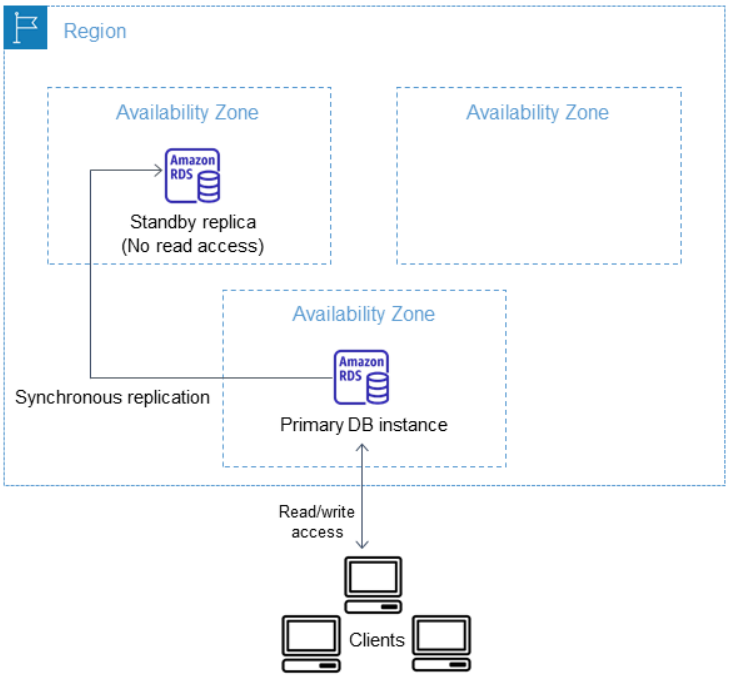
Avvertenze

Dobbiamo tenere presente che, come per le istanze "tradizionali", il numero di connessioni simultanee è determinato dalla dimensione della memoria RAM assegnata al cluster. Quando configuriamo un'istanza serverless, il numero di connessioni simultanee è determinato dal limite superiore delle ACU configurate. Se è necessario aumentare il numero di connessioni, dobbiamo cambiare il numero massimo di ACU e riavviare l'istanza per applicare le modifiche.

Alta Disponibilità: Multi-AZ, read-replicas. Facciamo chiarezza!

Anche se la documentazione spiega chiaramente le singole funzionalità, c'è sempre confusione riguardo le istanze RDS Multi-AZ, Cluster Multi-AZ e Read Replica Aurora. Affrontiamo questo problema!

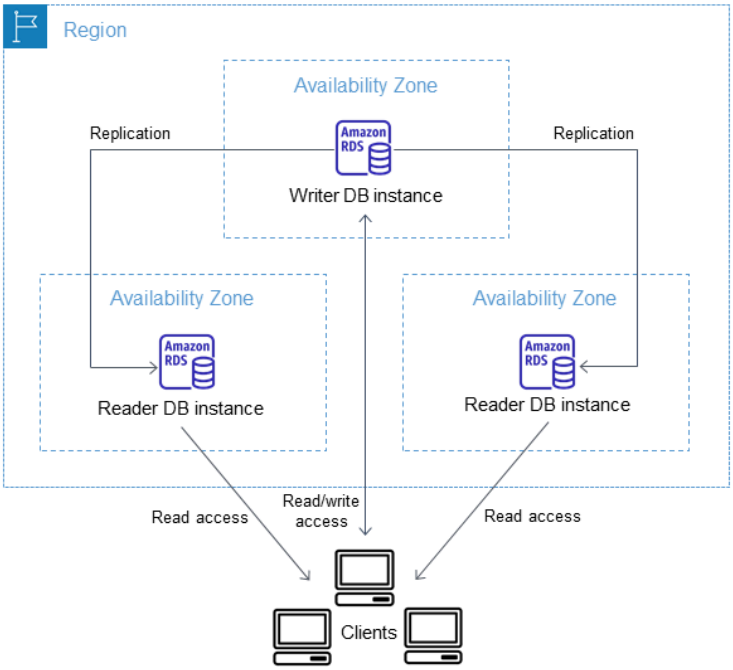
Prima di tutto, **un'istanza Amazon RDS Multi-AZ** è una copia sincrona da un'istanza primaria a una replica standby. **Non è possibile utilizzare l'istanza standby per leggere dati o per alleggerire il carico dato dalle letture**; è disponibile solo come opzione di failover in caso di problemi o di manutenzione. In caso di failover, il ripristino avviene in 60 - 120 secondi.



Source:

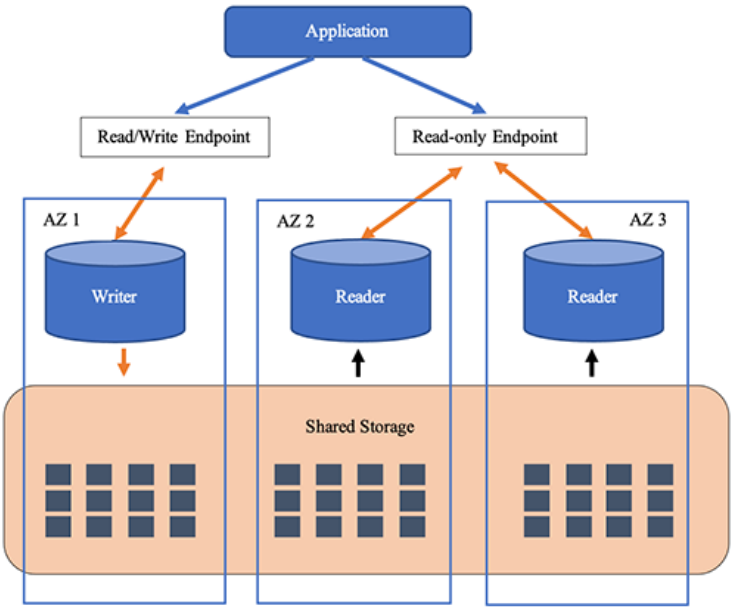
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Concepts.MultiAZSingleStandby.html

Possiamo poi creare cluster **Amazon RDS Multi-AZ**. In questo caso, **la replica è semi-sincrona** e utilizza la replica nativa del database engine. Si possono avere **fino a due istanze di sola lettura (read replica)** accessibili per distribuire il carico dato dalle letture. Tuttavia, con questa configurazione, ci potrebbe essere un ritardo nella replica, ma di contro la latenza di scrittura è inferiore rispetto a una istanza Multi-AZ. I failover vengono solitamente completati in meno di 35 secondi.



Source: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/multi-az-db-clusters-concepts.html

Come abbiamo detto prima, quando si usa il motore **Aurora** i dati vengono replicati automaticamente su tre Availability Zone. Possiamo aggiungere fino a 15 read replica per scalare le letture. In caso di fallimento, il failover avviene in meno di 30 secondi. Dobbiamo tenere presente che con un cluster Aurora Multi-AZ, l'applicazione deve utilizzare l'endpoint del cluster, gestito e aggiornato da Aurora.



Source: https://aws.amazon.com/blogs/database/improve-application-availability-on-amazon-aurora/

Consigli e trucchi per ridurre il tempo di failover

Amazon RDS Proxy può ridurre il tempo di failover in ogni caso preso in esame perché gestisce il pooling delle connessioni tra l'applicazione l'architettura database scelta. Inoltre, permette di ridurre il numero massimo di connessioni al database necessarie, alleviando così i picchi tipici dovuti alle applicazioni serverless. Dobbiamo tenere presente che Amazon RDS Proxy è un servizio che può influire sui costi!

Disponibilità e scalabilità globale: Amazon Aurora Global Database vs Amazon Aurora Postgres Limitless vs Amazon Aurora Distributed SQL

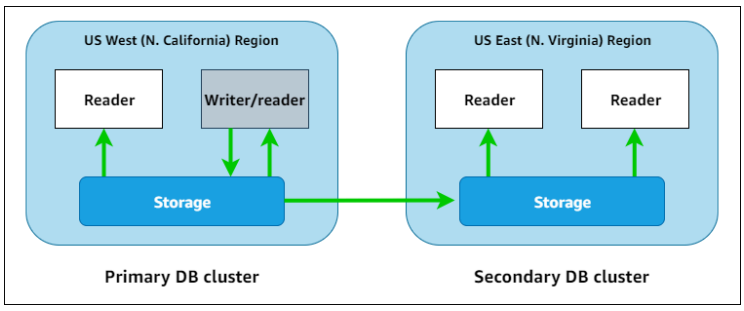
Scopriamo ora le versioni di Aurora per quanto riguarda gli ambienti distribuiti. Queste implementazioni si concentrano su aspetti diversi, ma potrebbero generare confusione per le loro simitudini.

Amazon Aurora Global Database

Amazon Aurora Global Database si concentra sulla **resilienza**: sfrutta infatti l'indipendenza tra i lo storage ed il compute per replicare i dati in una regione diversa. Quando si configura un'istanza Global Database, sarà definito un cluster primario (contenente un'istanza reader ed una writer) e un cluster secondario contenente solo read replica.

Il forwarding delle scritture (**write forwarding**) è supportato, quindi se viene effettuata una scrittura utilizzando l'endpoint del cluster secondario, Aurora se ne occuperà inoltrandola al cluster DB primario.

Se l'applicazione è sensibile alla latenza, dobbiamo tenere a mente che quando sono coinvolte lunghe distanze, la velocità della luce è un limite fisico che non può essere evitato (almeno al momento della stesura di questo articolo!).



Source: https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/aurora-global-database.html

Switchover, Managed Failover e Manual Failover: le differenze

Quando si tratta di **resilienza**, ci sono diversi meccanismi che permettono di eseguire test, programmare la manutenzione e, naturalmente, avere una rete di sicurezza in caso di problemi.

Con l'operazione di **switchover**, tutto è sotto controllo e i dati sono completamente replicati nel cluster secondario. Pertanto, l'RPO (Recovery Point Objective) è zero, senza perdita di dati. Questa operazione aiuta a eseguire manutenzioni programmate, test di failover, rotazione di region o ripristino da uno scenario di disastro nella region originale.

La seconda opzione disponibile è il **failover gestito (managed failover)**. In caso di problemi a livello di region o servizio, un failover gestito aiuta a mantenere la business continuity, passando dal cluster primario a quello secondario senza attendere la sincronizzazione dei dati. Ci sarà ovviamente perdita di dati, ma non verranno apportate modifiche alla topologia di replica (eccetto per lo switch di ruolo del cluster).

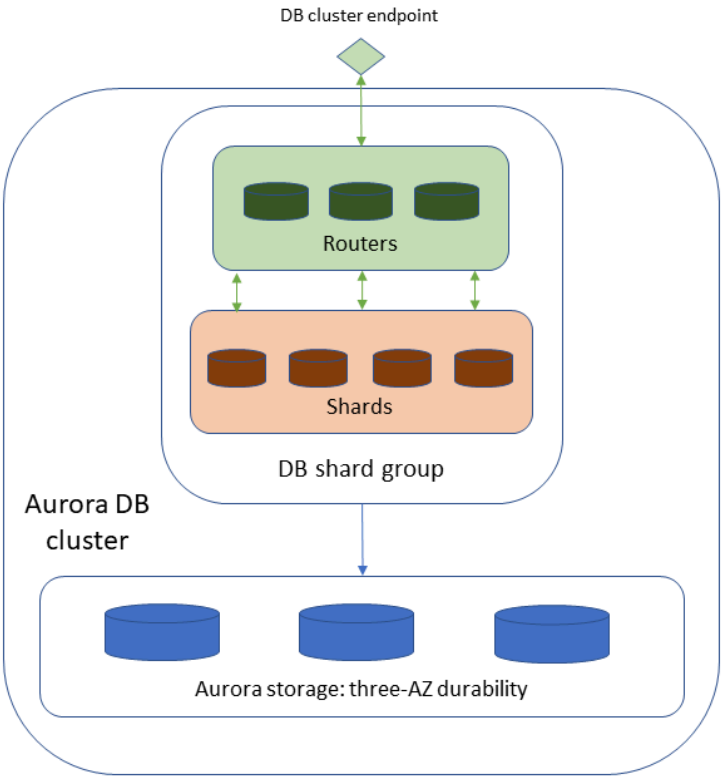
Il **failover manuale (manual failover)** è un'operazione critica; dobbiamo essere sicuri di quel che facciamo e capirne le implicazioni. Infatti, viene interrotta la replica del cluster esistente, passando ad un cluster standalone nella region secondaria e promuovendone una read replica a writer instance. Per ripristinare l'architettura dovremo ricreare la topologia di replica aggiungendo una region secondaria: non sono disponibili opzioni automatiche per il failback.

Consiglio bonus: se l'RTO non è molto vicino a zero, è possibile ridurre i costi sfruttando la **configurazione headless**; una volta fatto il deploy del cluster in una region secondaria basterà terminare la nuova istanza Aurora reader appena creata. La replica dei dati non si interromperà e, avviando una nuova istanza nel cluster secondario, questa si avvierà senza problemi.

Amazon Aurora PostgreSQL Limitless Database

Come suggerisce il nome, questa implementazione è attualmente disponibile solo per la compatibilità con il motore PostgreSQL. Si concentra sulla **scalabilità** utilizzando la tecnica dello sharding per distribuire i dati tra i nodi. Questo permette di scalare orizzontalmente, raggiungendo milioni di transazioni in scrittura e petabyte di dati.

La sua architettura differisce leggermente dalle altre perché, quando si fa sharding dei dati, occorre instradare le query ai nodi corretti per interrogare i dati. I nodi router possono ricevere query e inviarle allo nodo shard corretto. Questa scalabilità comporta un costo in termini di effort operativo: dobbiamo sapere come partizionare correttamente i dati e, poiché sono partizionati, non vogliamo che si verifichi il fenomeno dello "hot shard" che riceve più query rispetto agli altri.



Dovremo anche gestire i nodi shard e monitorarne il carico utilizzando le metriche CloudWatch. Per gestire il carico è possibile cambiare la capacità dei nodi shard, dividere gli shard in più gruppi di shard (ma attenzione: non è possibile unire gli shard) e aggiungere nodi router.

Avvertenze

Come suggerisce il nome, non è prevista la compatibilità con MySQL. Inoltre, Amazon Aurora PostgreSQL Limitless non è disponibile in tutte le region e non è possibile modificare le chiavi usate per lo shard (incluso il loro valore nelle righe delle tabelle); è necessario eliminarle e ricrearle. A questo indirizzo l'elenco completo delle limitazioni.

Infine, a questo indirizzo sono contenute le limitazioni per le query di definizione e manipolazione dati (DDL e DML).

Amazon Aurora Distributed SQL

Con il suo annuncio recente, Amazon Aurora Distributed SQL (DSQL) si concentra su **resilienza e scalabilità**. È un cluster HA multi-region active-active che scala automaticamente e con infrastruttura completamente gestita. Sebbene combini le funzionalità dei cluster Global e Limitless, al momento della stesura di questo articolo ha molte limitazioni. Attualmente è in anteprima solo in region selezionate. Diventerà più ricco di funzionalità in futuro e, per alcuni casi d'uso, potrebbe essere sufficiente per garantire la scalabilità e resilienza dell'applicazione senza ulteriore effort operativo.

Limitazioni di Aurora Distributed SQL

Attualmente, è disponibile solo PostgreSQL e ci sono limitazioni su oggetti disponibili e query. È possibile avere solamente un singolo database per cluster senza viste, trigger, tabelle temporanee o sequence. Le chiavi esterne non sono supportate e non è possibile eseguire TRUNCATE, VACUUM o creare funzioni utilizzando plpgsql (o qualsiasi altro linguaggio diverso da SQL standard).

Sebbene supporti operazioni ACID, le transazioni non possono avere operazioni DDL e DML miste e possono contenere al massimo una dichiarazione DDL.

TL;DR

- Amazon RDS e Amazon Aurora hanno architetture e caratteristiche diverse. È possibile scegliere tra un approccio più "tradizionale" utilizzando Amazon RDS (incluso RDS Custom per scenari specifici) o sfruttare il decoupling di storage e compute offerto da Amazon Aurora.
- In termini di alta disponibilità, Amazon RDS offre Multi-AZ per la replica sincrona verso un'istanza standby e cluster Multi-AZ con fino a due standby leggibili. Amazon Aurora replica automaticamente i dati su tre Availability Zone e consente fino a 15 read replica.
- Aurora Global Database replica i dati tra le regioni per disponibilità e scalabilità globali. Aurora PostgreSQL Limitless utilizza lo sharding per lo scaling orizzontale, e Aurora Distributed SQL combina cluster attivo-attivo multi-region con scaling automatico.
- Ci sono limitazioni per le architetture globali e lo scaling orizzontale; consulta l'articolo per un elenco più completo!

Che tipo di architettura state usando per i vostri database database? Ci sono esigenze particolari che avete dovuto affrontare? Fatecelo sapere nei commenti!

About Proud2beCloud

Proud2beCloud è il blog di beSharp, APN Premier Consulting Partner italiano esperto nella progettazione, implementazione e gestione di infrastrutture Cloud complesse e servizi AWS avanzati. Prima di essere scrittori, siamo Solutions Architect che, dal 2007, lavorano quotidianamente con i servizi AWS. Siamo innovatori alla costante ricerca della soluzione più all'avanguardia per noi e per i nostri clienti. Su Proud2beCloud condividiamo regolarmente i nostri migliori spunti con chi come noi, per lavoro o per passione, lavora con il Cloud di AWS. Partecipa alla discussione!

Damiano Giorgi

Ex sistemista on-prem, pigro e incline all'automazione di task noiosi. Alla ricerca costante di novità tecnologiche e quindi passato al cloud per trovare nuovi stimoli.L'unico hardware a cui mi dedico ora è quello del mio basso; se non mi trovate in ufficio o in sala prove provate al pub o in qualche aeroporto!
