Japan

Aurora/RDS MySQL Latest Update

Amazon Web Services G.K.

Disclaimer

- This document describes the service content and pricing as of January 3, 2022. Please refer to the AWS official website (http://aws.amazon.com/) for the latest information

- AWS does not offer binding price quotes. AWS pricing is publicly available and is subject to change in accordance with the AWS Customer Agreement available at http://aws.amazon.com/agreement/. Any pricing information included in this document is provided only as an estimate of usage charges for AWS services based on certain information that you have provided. Monthly charges will be based on your actual use of AWS services, and may vary from the estimates provided

Agenda

- Amazon Aurora MySQL version3
- Amazon Aurora Serverless v2 (Preview)
- Amazon RDS Multi-AZ DB cluster (Preview)
- Summary

Amazon Aurora MySQL version3

Amazon Aurora

MySQL and PostgreSQL compatible relational database built for the cloud

Performance and availability of commercial databases at 1/10th the cost

Performance and scalability

Several times faster than standard MySQL and PostgreSQL

15 read replicas

Availability and durability

Fault-tolerant self-healing storage

Six copies of data across three AZs

Single Global database with cross-region replication

Highly secure

Network isolation

Encryption at rest/transit

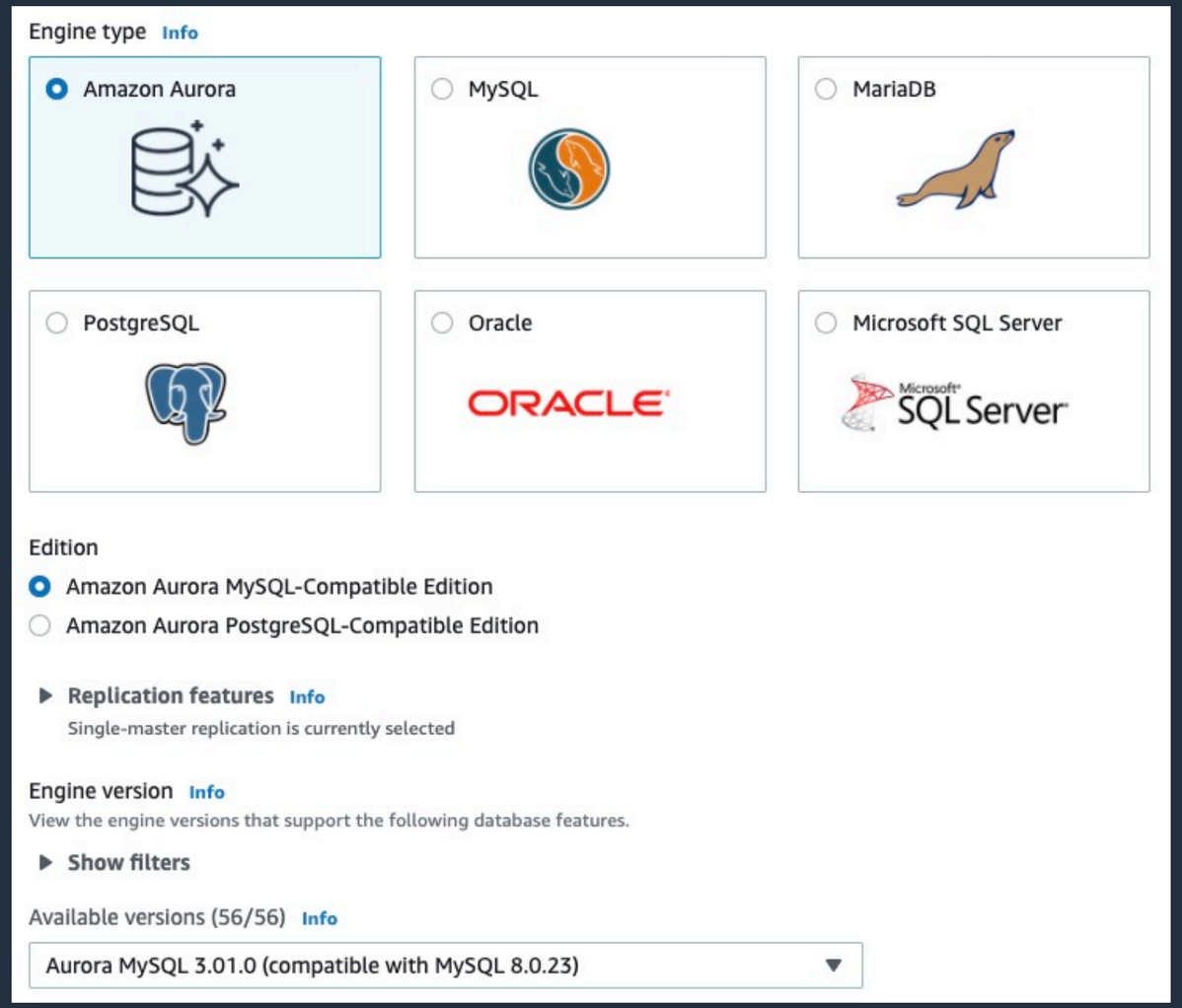
Fully managed

Managed by RDS: no hardware provisioning, software patching, setup, configuration, or backups

The fastest growing service in the history of AWS

Introducing Aurora MySQL version3

- Aurora MySQL version3は、Community MySQLから、共通テーブル式(CTE)サポート、ロールベース認証、レプリケーションの強化、ウィンドウ関数、インスタントDDLなどサポート
- Aurora MySQL version3は、MySQL 8.0.23 Community Editionとの互換性があり、Auroraがサポートされているすべてのリージョンで利用可能
- Aurora MySQL 3.01.0 は、Aurora MySQL2.10.0 までのAurora特有のバグフィックスを含む



Current Aurora MySQL version options

Major Versions

Aurora MySQL Version	Community Major Version	Aurora version EOL no earlier than this date
1	MySQL 5.6	September 30, 2022
2	MySQL 5.7	February 29, 2024
3 **(NEW)**	MySQL 8.0	

Minor Versions

Aurora MySQL Version	Community Minor Version
1.x	5.6.10a
2.x	5.7.12
3.x	各Auroraマイナーバージョンは、コミュニティMySQL 8.0マイナーバージョンにマッピングされる (e.g., 3.01 → 8.0.23)

Aurora MySQL version3から、MySQL Community Edition のマイナーバージョンリリースにより近い形でリリースを行う

Aurora MySQL version3 の各マイナーバージョンは、対応する MySQL 8.0 Community Edition のマイナーバージョンと互換性がある

Overview of community MySQL 8.0 and enhancements

Instant DDL

- Supersedes “Lab Mode” Fast DDL
- Compatible with the instant DDL from community MySQL 8.0
- ALGORITHM=INSTANT with the ALTER TABLE statement
- Only modifies metadata in the data dictionary
  - No exclusive metadata locks
- Not all operations are supported

https://dev.mysql.com/doc/refman/8.0/en/innodb-online-ddl-operations.html

- Limitations*
  - No temp table support
  - No compressed row format support
  - Only adds last column

*Inherited from MySQL and apply to both Aurora and MySQL Community

Common table expressions

- A named temporary result set
- Created using WITH clause
- Useful for hierarchical data traversal, avoiding repeated sub-queries, better readability
- Exists within the scope of a single statement and can be referred within that scope

```
WITH cte1 AS (SELECT a, b FROM
table1)

SELECT a FROM cte1;
```

- Can refer to itself or other CTEs. Recursive CTEs when self-referred.

```
WITH RECURSIVE cte AS
(
SELECT category_id, name FROM category WHERE parent IS NULL
  UNION ALL
SELECT c.category_id, c.name FROM category c JOIN cte
  ON cte.category_id=c.parent
)
SELECT name FROM cte;
```

Window functions

- Aggregate-like operations, aggregate over a rolling window and output results per row.
- Supports RANK(), DENSE_RANK(), NTILE(), ROW_NUMBER() and more.

```
SELECT dept, empid, salary, rank() OVER (PARTITION BY dept
ORDER BY salary DESC) FROM employee;
 dept     | empid | salary | rank
----------+-------+--------+------
 IT       |     8 |   6000 |    1
 IT       |    10 |   5200 |    2
 IT       |    11 |   5200 |    2
 IT       |     9 |   4500 |    4
 IT       |     7 |   4200 |    5
 HR       |     2 |   3900 |    1
 HR       |     5 |   3500 |    2
```

Role-based access control

- ロールとは、特権の名前付きコンテナ
- Aurora MySQL version3からサポート
- ロールの作成と削除、ロールへの権限の付与と剥奪が可能
- ユーザーグループへの権限付与・剥奪を効率化し、ユーザー管理を簡素化
- セッション中にSET ROLEでロールを設定／変更可能
- SHOW GRANTSでアクセス権を表示

New index types

- New index types – Invisible, Descending, Functional key parts
- **Invisible index** -
  - ALTER TABLE … INVISIBLE/VISIBLE
  - オプティマイザでだけ利用されない
  - スキーマの最適化に有効
- **Descending index** –
  - DESCキーワードを使用
  - 値は降順ソートで格納される
- **Functional index**–
  - 列の値ではなく、式の値をインデックスする
  - 他の方法ではインデックスを作成できない値のインデックス作成を可能に

Binary log enhancements

他のAurora MySQLのバージョンと同様に、バイナリログのレプリケーションはデフォルト無効。バイナリログは有効にすることができ、Aurora MySQL version3ではいくつかの改良が加えられている

Multi-threaded replication (MTR)

- 特にプライマリDBインスタンスに高い割合で書き込みが発生するワークロードの場合、バイナリログレプリケーションのパフォーマンスを向上
- レプリカの変更を並行して適用するため、シングルスレッドレプリケーションと比較してレプリケーションの遅延が減少
- レプリカDBクラスタのreplica_parallel_workersに0より大きい値を設定することで有効化

Replication filtering

- レプリケーションフィルターを使用し、どのデータベースとテーブルをレプリケートするか(どのデータベースとテーブルを含むか、またはどのテーブルを除外するか)を指定
- replicate-do-*とreplicate-ignore-*のフィルタリングパラメータを使用

Binary log transaction compression

- zstd アルゴリズムがトランザクションのペイロードを圧縮するために使用される。圧縮されたトランザクションは、その後バイナリログに書き込まれる。トランザクションは、転送中およびバイナリログレプリカ上で圧縮される
- プライマリとレプリカの両方のAurora MySQLクラスタ上でディスクスペースを節約
- バイナリログがネットワーク帯域幅をより少なく消費し、トランジットパフォーマンスを向上

And more…

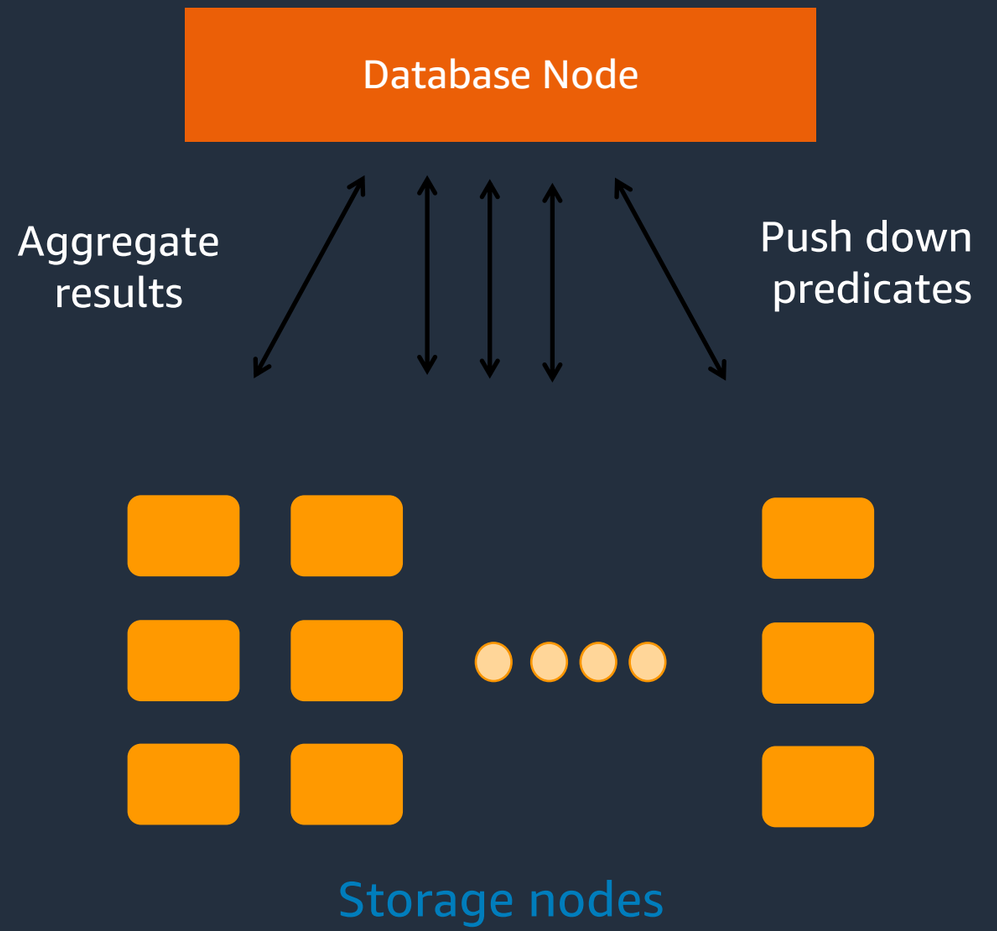
- JSON functions
- NOWAIT and SKIP LOCKED clauses
- Hints

Aurora specific feature enhancements with Aurora MySQL version3

New Parallel Query features

Aurora storage has thousands of CPUs

- クエリ処理のプッシュダウンと並列化
- データの近くに処理を移すことで、ネットワークトラフィックとレイテンシーを削減



New features with Aurora MySQL version3

Support for:

- カラムのデータ型。TEXT、BLOB、JSON、GEOMETRY、VARCHARおよびCHAR
- パーティションで区切られたテーブル
- 集計関数、GROUP BY句、HAVING句

Currently unsupported features on Aurora MySQL version3

- **Query cache**
  - Aurora MySQL version3 から削除
- **In place upgrades**
  - Version 2からVersion3へのin-placeアップデートは今後サポート予定
- **Backtrack**
  - 今後サポート予定
- Restoring a physical backup from **Percona XtraBackup** to an AMS 3 cluster
  - 今後サポート予定

Behavioral changes in Aurora MySQL version3

Behavioral changes

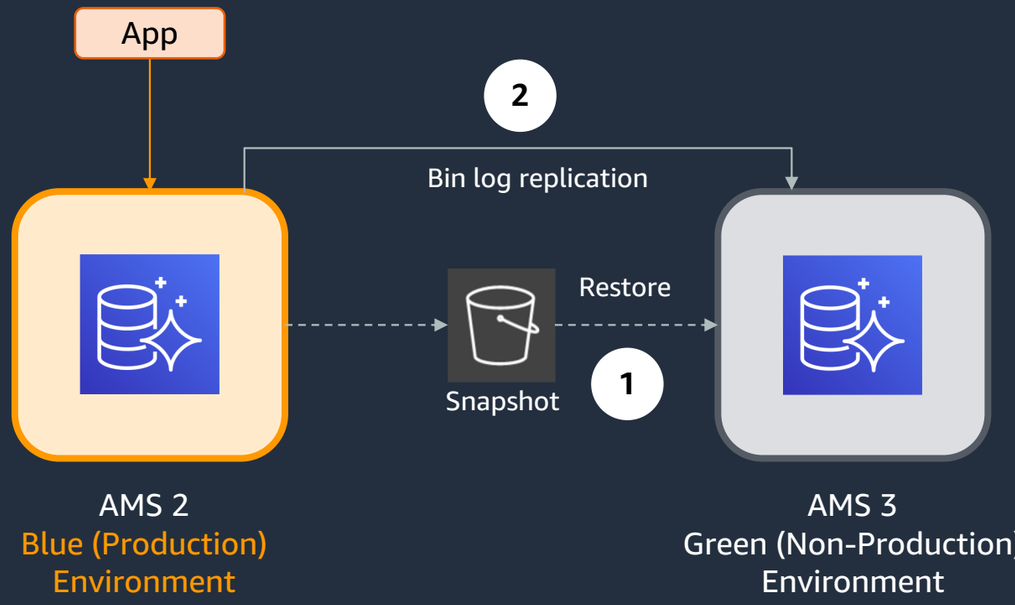
- **Temp Tables:** 新しいTempTableストレージエンジンがデフォルトで使用されまる。メモリオーバーフロー時は、InnoDBまたはメモリマップドファイル、あるいはその両方を使用
- **Data Dictionary:** メタデータファイル、非トランザクションテーブル、ストレージエンジン固有のデータディクショナリを使用していた以前のバージョンとは異なり、新しいトランザクションデータディクショナリを使用。共有グローバルディクショナリキャッシュを導入
- **AUTO_INCREMENT:** 再起動後も保持
- **Error codes:** いくつかのエラーコードが削除された。エラー処理に特定の MySQL エラーコードを使用している場合は見直しが必要
- **ASC/DESC clause for GROUP BY:** GROUP BY句のASCまたはDESCの修飾子が削除。代わりにORDER BY句を使用

Migration path to Aurora MySQL version3

Upgrade to Aurora MySQL version3 from Aurora MySQL 1 or 2

Aurora MySQL 2 → 3: Aurora MySQL 2クラスタをAurora MySQL 3にアップグレードするには、Aurora MySQL 2クラスタのスナップショットを作成し、スナップショットをリストアして、新しいAurora MySQL 3.クラスタを作成

ソースクラスタとターゲットクラスタ間のバイナリログレプリケーションを有効にし、ダウンタイムを最小に

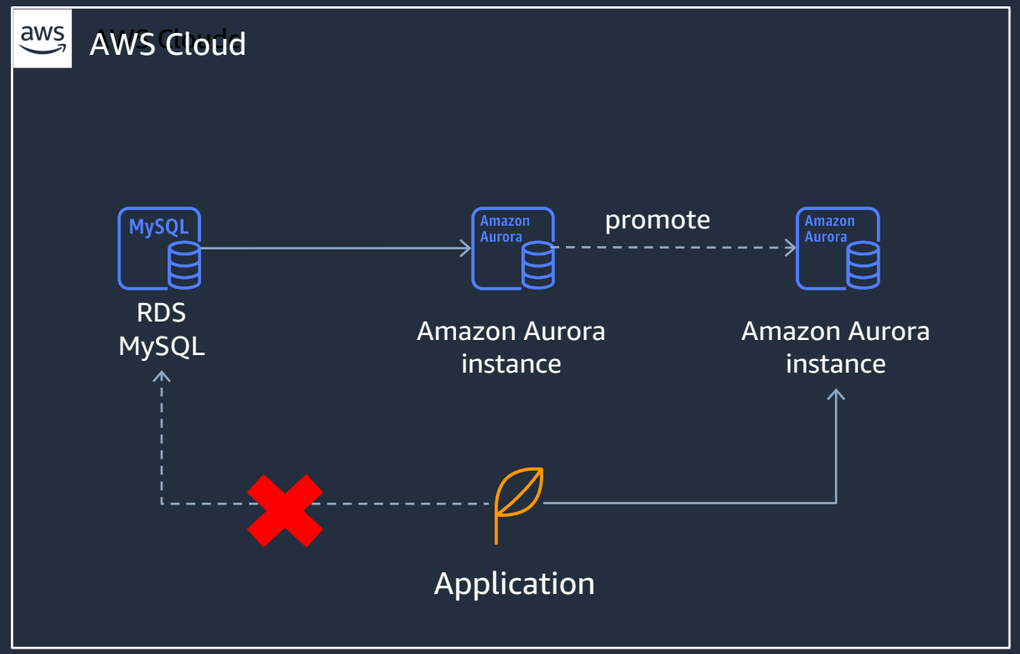


Aurora MySQL1 → 3: Aurora MySQL1からアップグレードするには、インプレースアップグレードオプションまたはスナップショット復元を使用して、 Aurora MySQL 2 への中間アップグレードを行う必要がある

Supported migration paths from RDS MySQL to Aurora MySQL version3

Aurora Read Replica: Amazon RDS for MySQL 8.0 プライマリ DB インスタンス(MySQL8.0.23 以降)から Aurora MySQL 3 リードレプリカ DB クラスターを作成

- RDS for MySQL DBインスタンスの全データを含むAMSクラスタを自動的に作成
- 自動化されたバイナリログレプリケーションにより、ダウンタイムがほぼゼロになる移行を実現
- 同一AWSアカウント、同一リージョン内のAuroraへの移行が可能



RDS Snapshot Migration: MySQL 8.0 (MySQL 8.0.23 以下) のスナップショットからRDSをAurora MySQL version3クラスタにリストア可能

バイナリログレプリケーションと併用することで、ダウンタイムがほぼゼロになる移行が可能

別アカウント、別リージョンへの移行も可能

Remember…

メジャーバージョンアップの前には必ずテストを行ってください!

- Aurora MySQL DB クラスタをAurora MySQL version3 にアップグレードする前に、アップグレードのテストを実施し、指摘項目に対処する
- mysqlcheck --check-upgrade コマンドを使用して評価を実行
- 本番のAuroraデータベースクラスタをアップグレードする前に、Aurora MySQL version3を利用しアプリケーションの動作や互換性をテスト

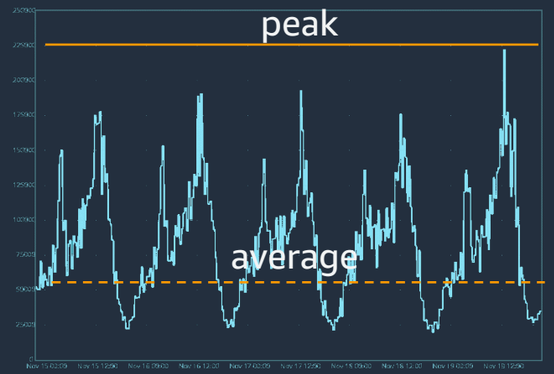
Other considerations before moving to Aurora MySQL version3

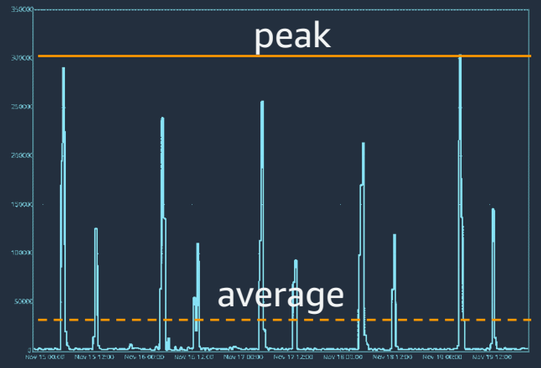
- ワークロードの将来性と最新機能へのアクセスを優先する場合はAurora MySQL version3検討
- Aurora MySQL version3 の長期サポート(LTS)バージョンは現在未サポート
  - ソフトウェアアップデートがセキュリティとバグフィックスに限定されることが保証される、LTSバージョンのデータベースエンジンが必要な場合は、Aurora MySQL2.x (MySQL 5.7 互換)を検討
- Aurora MySQL1.x (MySQL 5.6互換)の利用は非推薦
  - 現在Aurora MySQL1.xを使用している場合、既存のデータベースをより最新バージョンのAurora MySQLにアップグレードすることを検討する

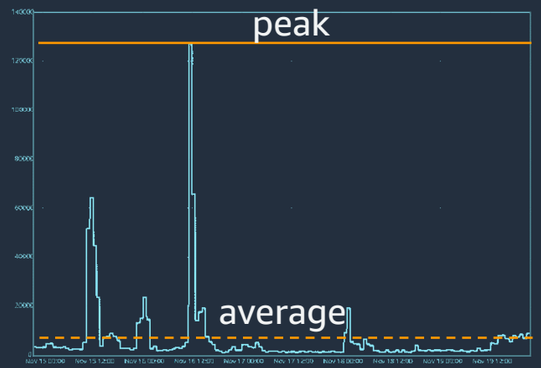
Amazon Aurora Serverless v2 (Preview)

Database capacity: cost vs. management

Variable and unpredictable workloads







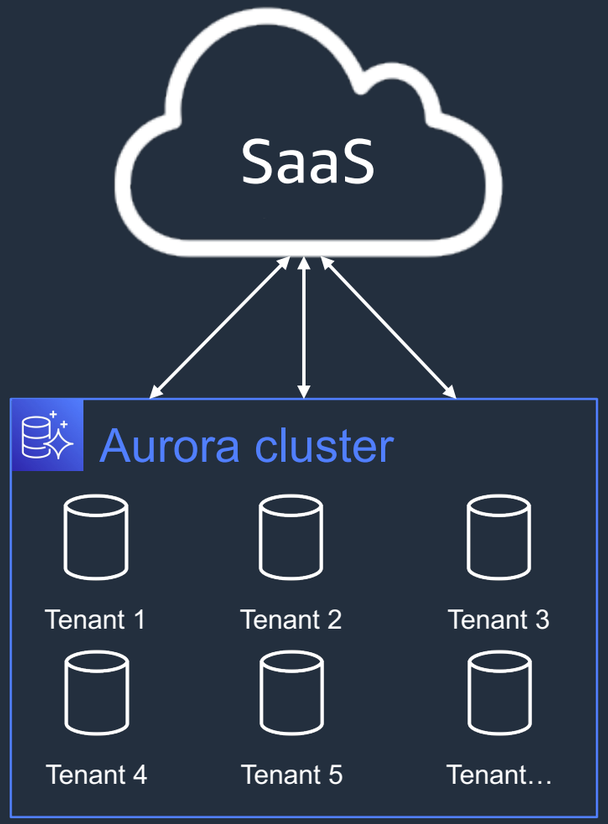
Insufficient capacity → Experience degradation

Provision for peak → Expensive

OR

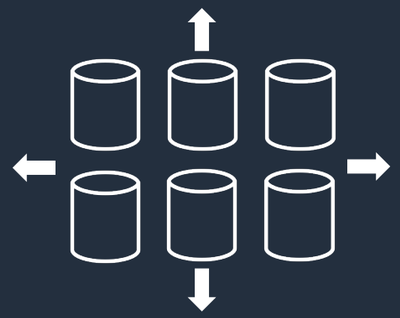
Continuously monitor and scale → Difficult, requires experts, involves downtime

Multi-tenant applications with per-tenant databases



- 数千のエンドカスタマーが、それぞれ独自のデータベースを保有
- データベースの共同利用による利用率の向上とコスト効率化
- バックアップやリストアなど、運用管理の粒度をトレードオフ
- ワークロードの繁忙による継続的な監視とシャッフルが必要

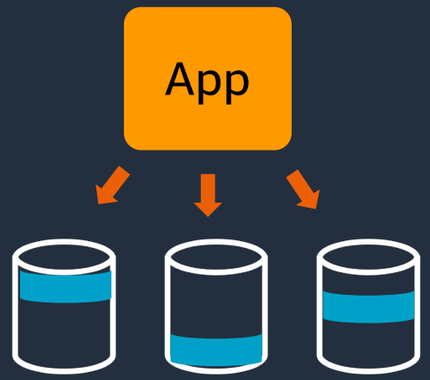
Several application backed by databases



100s to 1,000s of databases

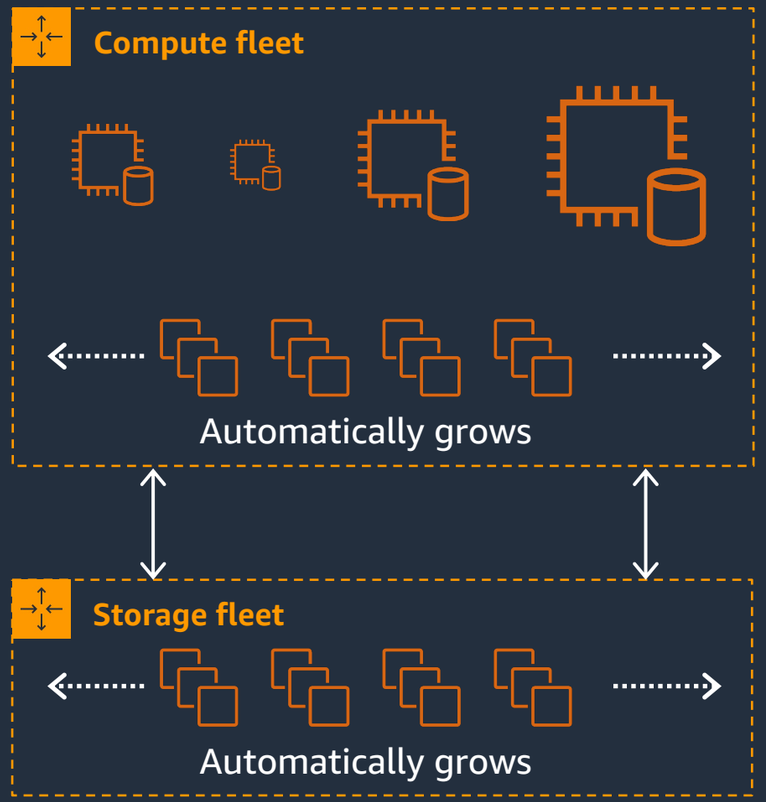
- 数百、数千のアプリケーションをお持ちのお客様
- 各アプリケーションは1つまたは複数のデータベースでバックアップされている
- アプリケーションの要件が変化するため、データベースの容量調整が必要
- データベース群の容量を予算内で管理するのは難しい

Scaled out databases



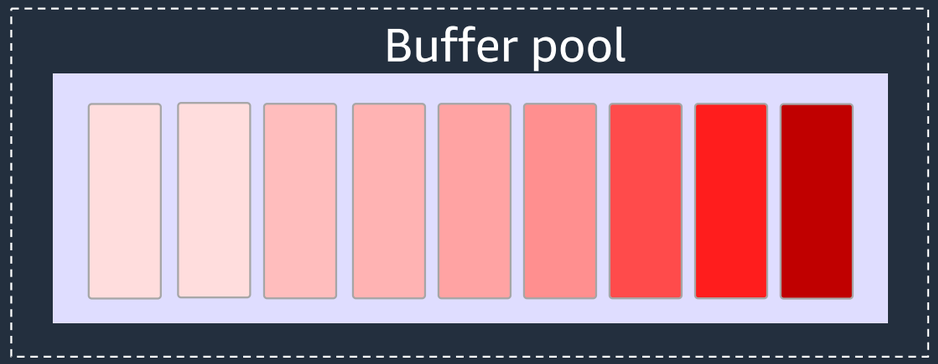
- 書き込みのスケーラビリティが必要なアプリケーションでは、データベースを複数のノードに分割し、高いスループットを実現
- 各ノードの容量を予測するのは困難で非効率的
- ノード数が少なすぎると、データの再分配が必要になり、ダウンタイムが発生
- ノードが多すぎると、すべてのノードが均等に使用されないため、高いコストがかかる

Instant, in place scaling



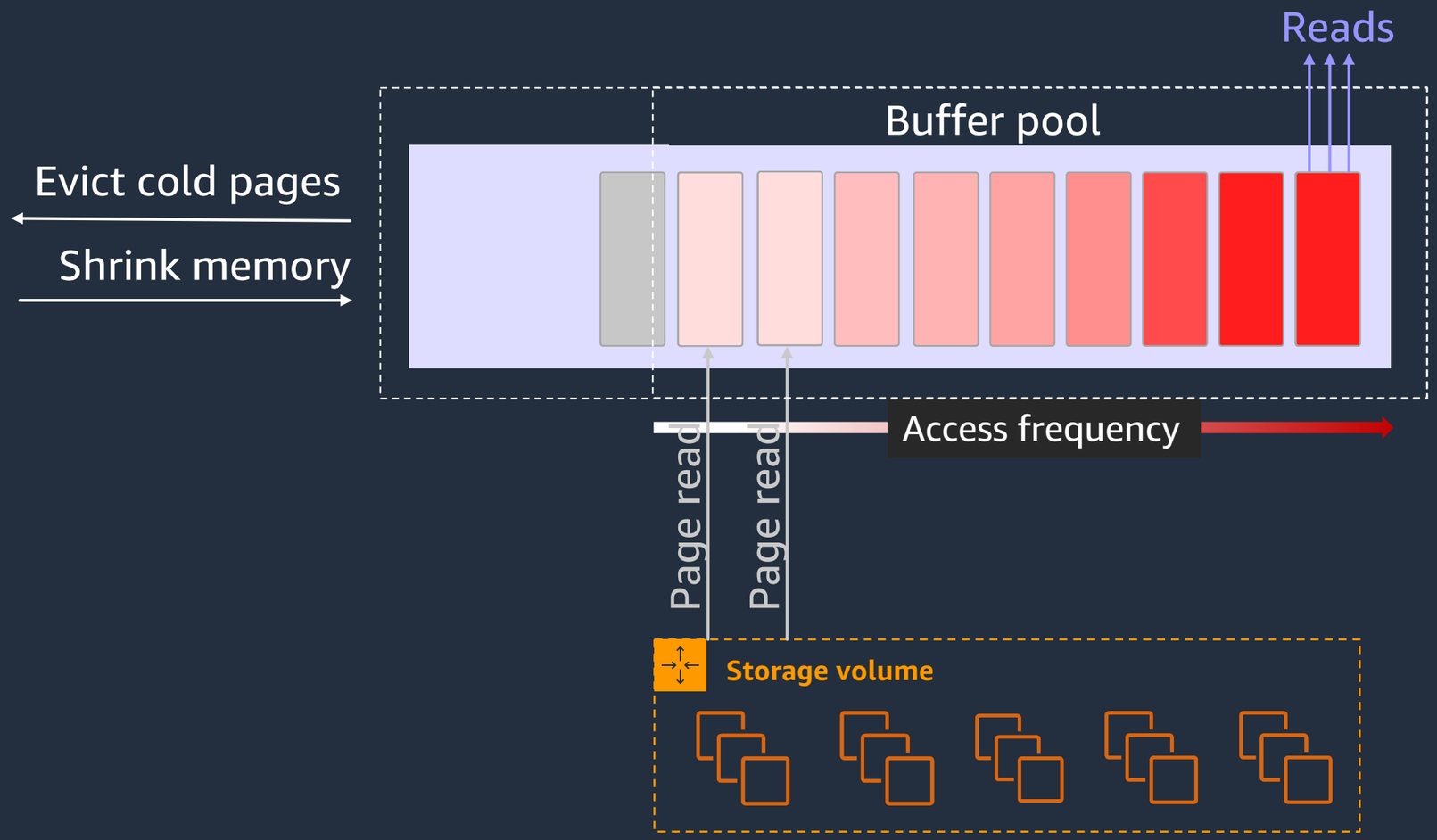
- CPUとメモリのリソースを追加することで、1秒以内にスケーリングが可能
- 数十万トランザクションを実行しても、スケーリングによる影響はない
- コンピュートフリートを継続的に監視し、スケールアップすることでホットスポットを管理
- バッファプールやコネクションなどの状態を保持したまま、アイドル状態のインスタンスをバックグラウンドで移動させることが可能
- 最大15倍の高速なスケールダウン

Buffer pool resizing



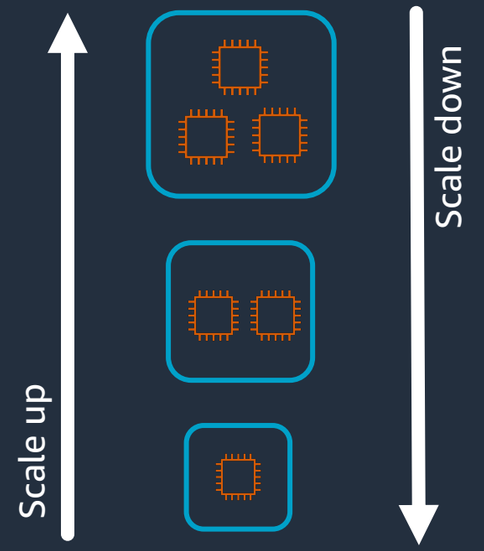
- バッファプールサイズは、データベース容量に合わせてスケーリングされる
- Parameters automatically adjusted:
  - MySQL: innodb_buffer_pool_size
  - PostgreSQL: shared_buffers
- Memory allocation: 75% for buffer pool and 25% for heap
- Buffer pool scaled down LFU(least frequently used)アルゴリズムとLRU(least recently used)アルゴリズムの組み合わせ

Buffer pool resizing

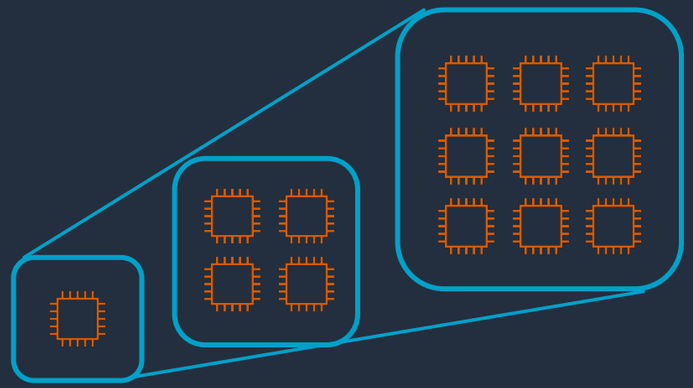


Fine-grained capacity adjustments

- Aurora Serverlessの容量は、Aurora Capacity Unit (ACU)で測定
- 1ACUは2GiBメモリ
- プロビジョニングされたAuroraインスタンスで利用可能なものと同様のCPUとネットワーク
- 0.5 ACU (1 GiB)の小さな容量からスタート可能
- 0.5 ACU単位でのきめ細かなスケーリング可能



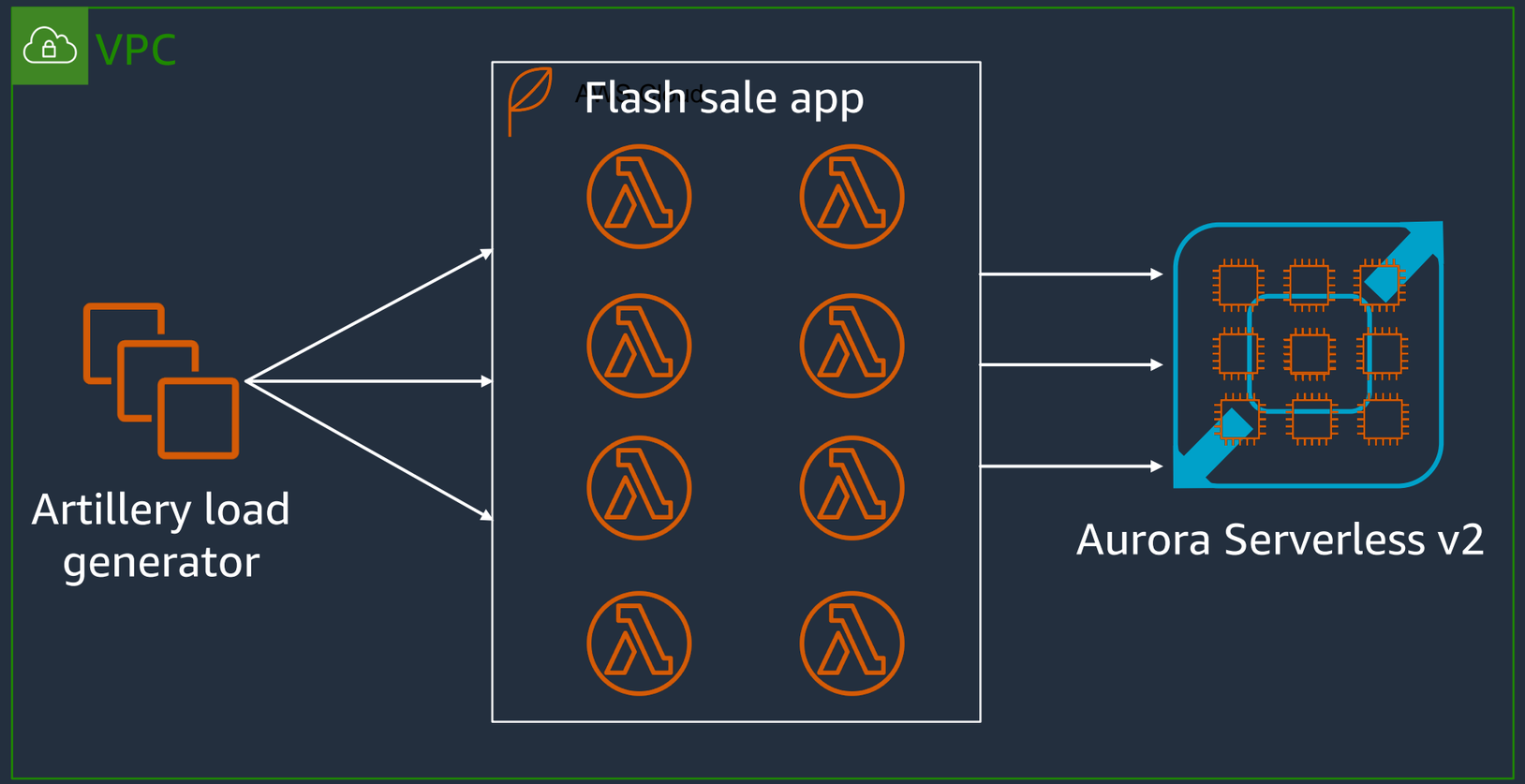
Scaling factors



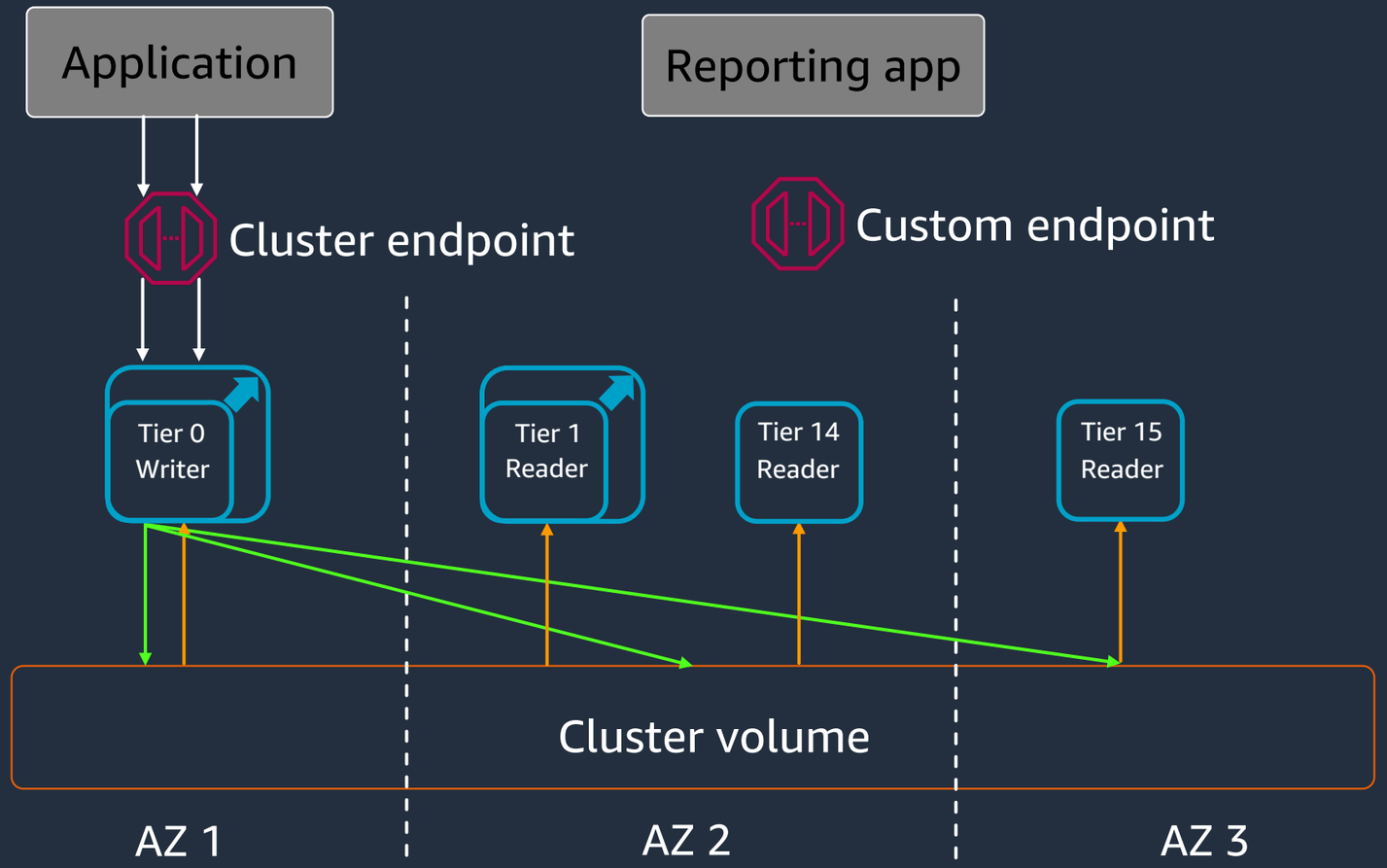
- スケールアップの速度は予測可能で、現在の容量に比例する - 大きなインスタンスはより速くスケールアップ
- フォアグラウンドおよびバックグラウンドプロセスのCPU使用率（例：パージやバキュームなど）
- 内部データ構造（バッファプールなど）のメモリ使用率
- ネットワークスループットはキャパシティに比例し、ネットワークスループットの必要性に応じて拡張される

Demo: Creating an Aurora Serverless v2 database

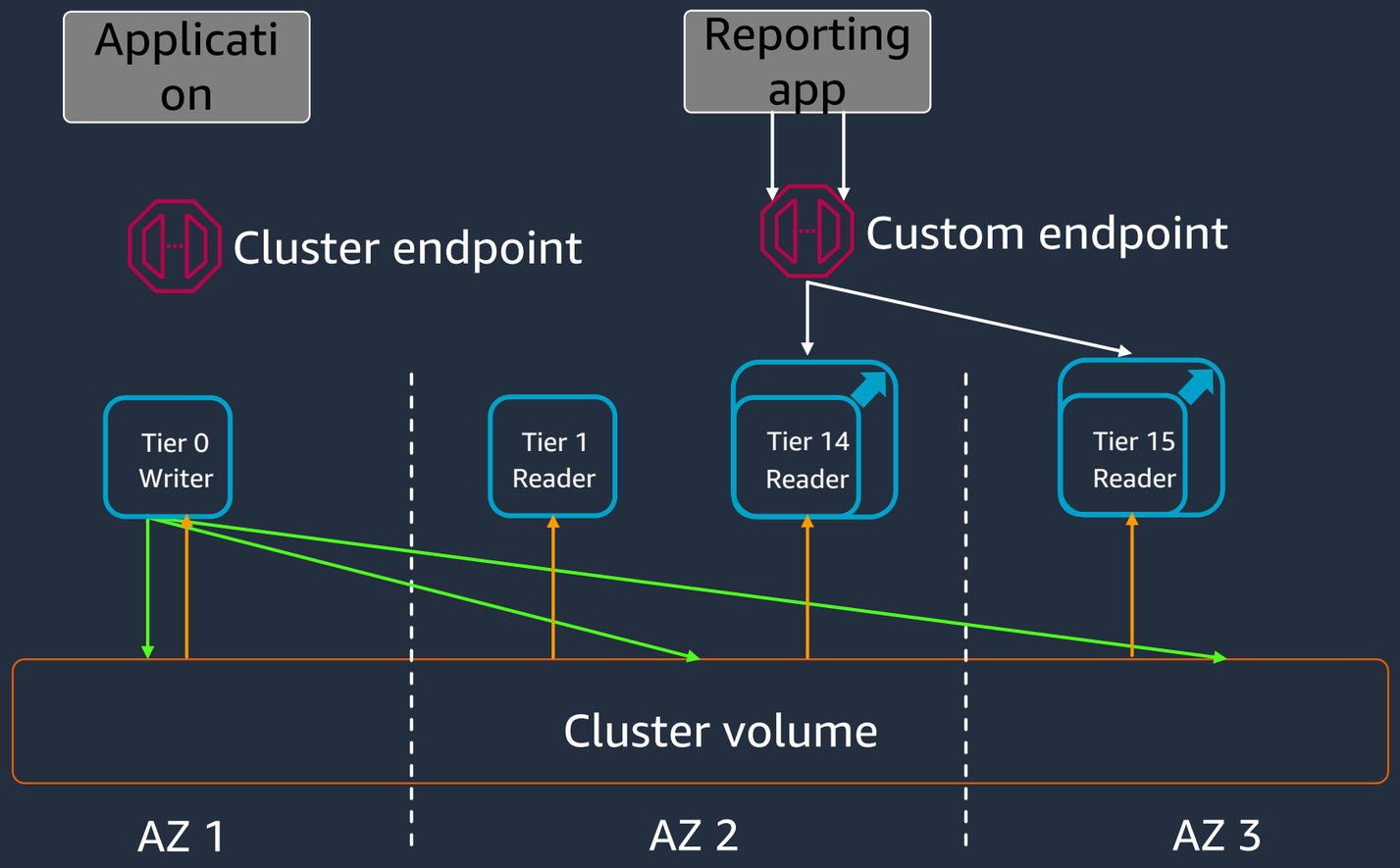
Demo: Flash sale



Multi-AZ high availability

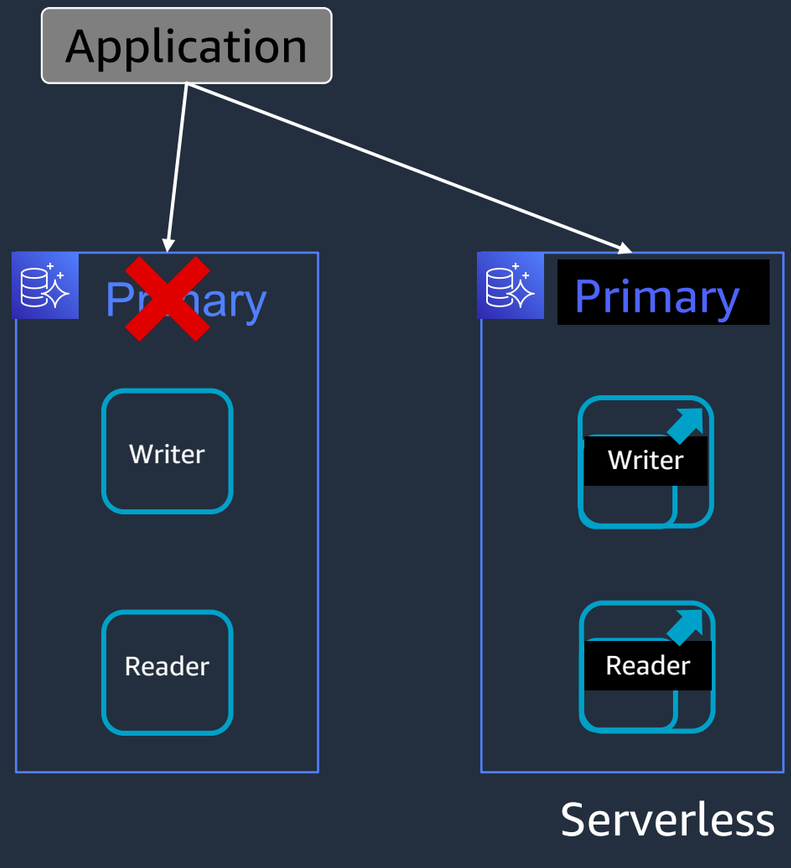


Read scalability



Serverless Global Database

- Aurora Serverless v2によるセカンダリリージョンの構成
- アイドル時の最小限の容量のみ課金
- フェイルオーバ後、ライタとリーダはスケールアップ
- エンドユーザに近い場所で読み取りを行うために、セカンダリリージョンを拡張



Supported major versions

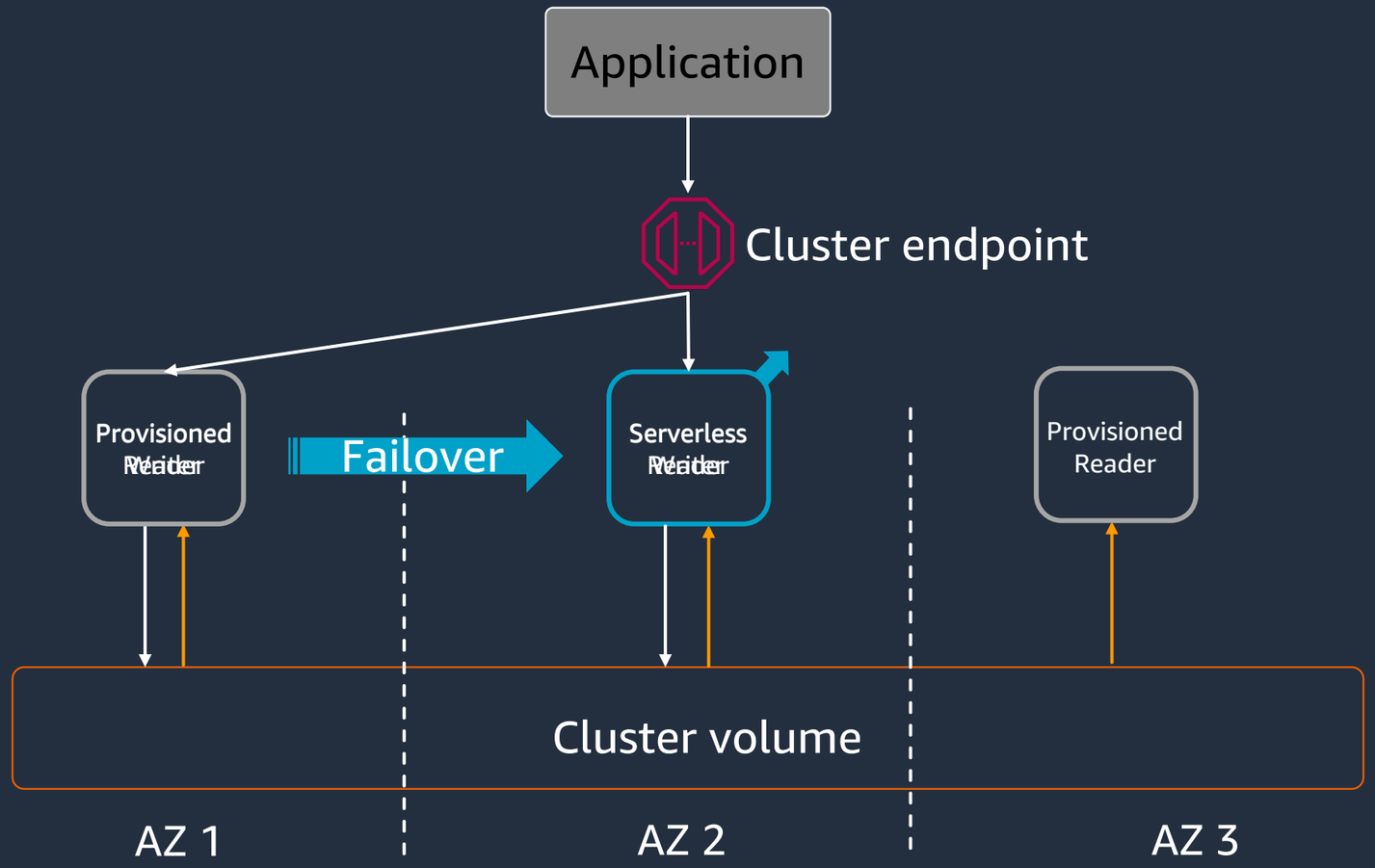
MySQL 8.0

- Windowing functions
- Atomic and online DDL
- SKIP LOCKED and NOWAIT options
- Improved JSON functionality

PostgreSQL 13

- De-duplication in b-tree indexes
- Improved partitioned table performance
- Incremental sorting for faster sorts
- Parallelized vacuum for indexes

Aurora clusters in mixed configuration



Demo: Mixed configuration of provisioned and serverless instances

Amazon RDS Multi-AZ DB cluster (Preview)

Amazon RDS Multi-AZ deployments

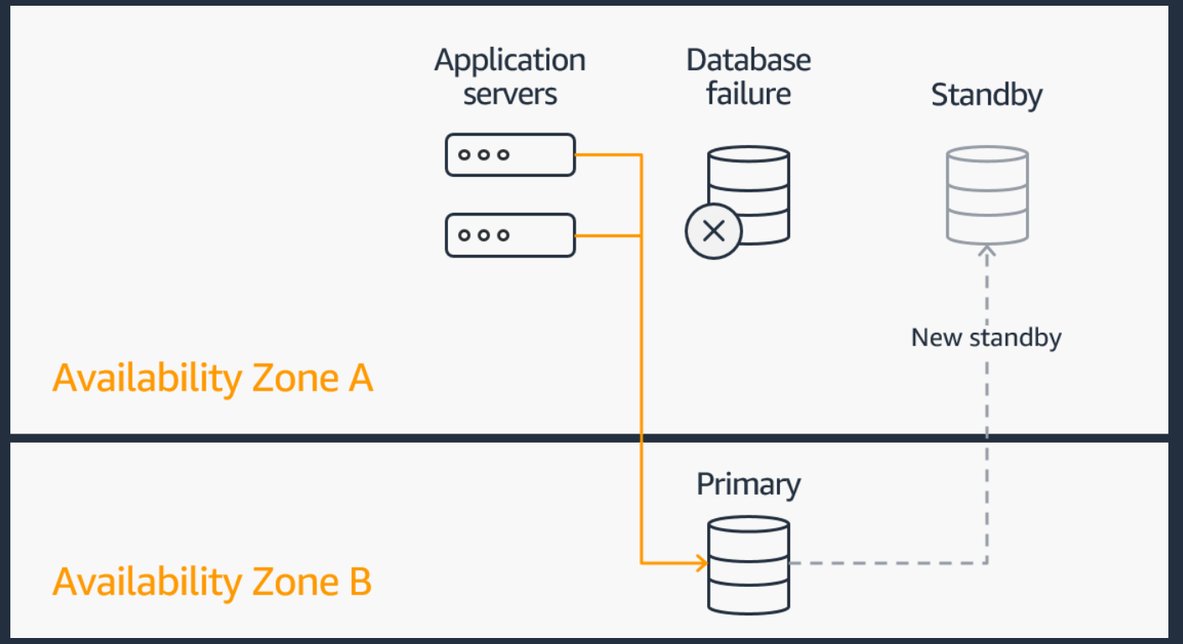
- 本番データベースのワークロードに最適
- Amazon RDSデータベースへ強化された可用性と耐久性を提供
- DBインスタンスのMulti-AZ配置を設定することで、Amazon RDSは自動的にプライマリDBインスタンスを作成し、異なるアベイラビリティーゾーンのスタンバイDBインスタンスにデータをレプリケート
- インフラストラクチャに障害が発生した場合、Amazon RDSはスタンバイへ自動フェイルオーバを実行し、フェイルオーバが完了するとすぐにデータベースへアクセス可能
- フェイルオーバ後もDBインスタンスのエンドポイントは変化しないため、アプリケーションは手動で設定変更の必要はなく、データベースアクセスを再開可能

Amazon RDS Multi-AZ deployments

Amazon RDS Multi-AZ deployments

Amazon Elastic Block Storeのストレージレベルで同期し、プライマリDBインスタンスとスタンバイDBインスタンスを構成

2010年より提供



Supported instances

Instance class	vCPU	Memory (GiB)	VPC only	EBS optimized	Max. bandwidth (mbps)	Network performance
db.m6gd						
db.m6gd.16xlarge	64	256	Yes	Yes	19,000	25 Gbps
db.m6gd.12xlarge	48	192	Yes	Yes	13,500	20 Gbps
db.m6gd.8xlarge	32	128	Yes	Yes	9,000	12 Gbps
db.m6gd.4xlarge	16	64	Yes	Yes	4,750	Up to 10 Gbps
db.m6gd.2xlarge	8	32	Yes	Yes	Up to 4,750	Up to 10 Gbps
db.m6gd.xlarge	4	16	Yes	Yes	Up to 4,750	Up to 10 Gbps
db.m6gd.large	2	8	Yes	Yes	Up to 4,750	Up to 10 Gbps
db.r6gd						
db.r6gd.16xlarge	64	512	Yes	Yes	19,000	25 Gbps
db.r6gd.12xlarge	48	384	Yes	Yes	13,500	20 Gbps
db.r6gd.8xlarge	32	256	Yes	Yes	9,000	12 Gbps
db.r6gd.4xlarge	16	128	Yes	Yes	4,750	Up to 10 Gbps
db.r6gd.2xlarge	8	64	Yes	Yes	Up to 4,750	Up to 10 Gbps
db.r6gd.xlarge	4	32	Yes	Yes	Up to 4,750	Up to 10 Gbps
db.r6gd.large	2	16	Yes	Yes	Up to 4,750	Up to 10 Gbps

RDS Multi-AZ DB cluster deployments (Preview)

- 読み取り専用のエンドポイントを使用して、読み取り可能なスタンバイDBインスタンスに接続可能
  - フェイルオーバ時に自動的に、読み取り/書き込みのトラフィックを制御することが可能

- 高い可用性と耐久性
  - より多くの読み取り性能が必要な場合
  - 一貫したトランザクション・レイテンシ

- Amazon RDSは、3つのアベイラビリティーゾーンにわたって1つのプライマリDBインスタンスと2つの読み取り可能なスタンバイDBインスタンスをプロビジョニングし、レプリケーションを自動的に設定

RDS Multi-AZ DB cluster deployments (Preview)

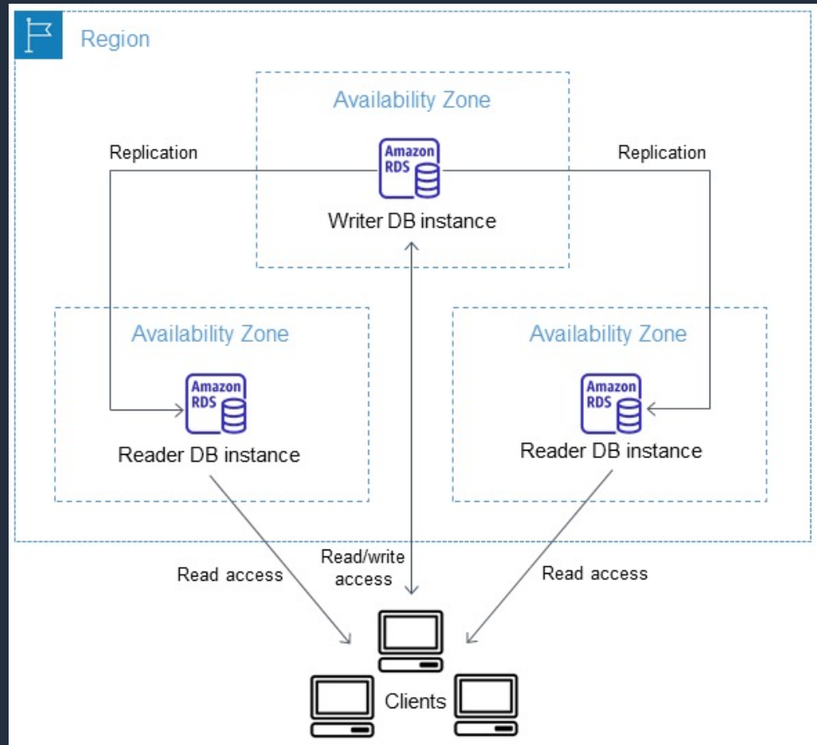
読み取り可能なスタンバイ・インスタンス

- NVMe SSDストレージとAmazon EBS
- 耐久性を損なうことなく、トランザクションコミットレイテンシを短縮
- AWS Gravtion2のインスタンスタイプを利用可能
  - R6gd
  - M6gd
- レプリケーションはデータベースエンジン標準の機能を利用

RDS Multi-AZ DB cluster deployments (Preview)

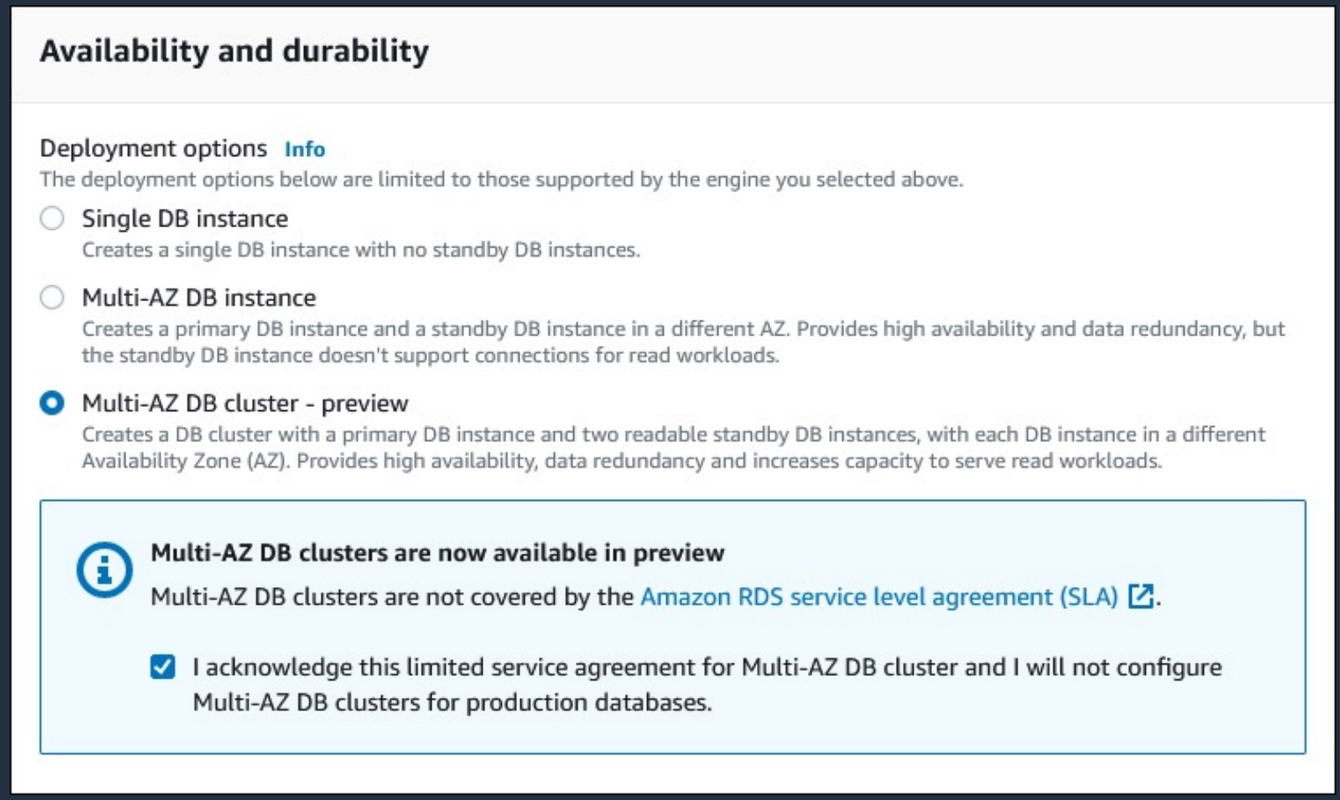
ライタDBインスタンスで変更が行われ、各リーダDBインスタンスに送られる

- 変更がコミットされるには、少なくとも1つのリーダDBインスタンスからのackが必要



Creating a DB cluster

Multi-AZ DB Cluster – previewを選択



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

Creating a DB cluster

CLI

create-db-clusterオプション

```
aws rds create-db-cluster \
   --db-cluster-identifier mysql-multi-az-db-cluster \
   --engine mysql \
   --engine-version 8.0.26  \
   --master-user-password password \
   --master-username databaseadmin \
   --port 3306 \
   --backup-retention-period 7  \
   --db-subnet-group-name default \
   --allocated-storage 4000 \
   --storage-type io1 \
   --iops 10000 \
   --db-cluster-instance-class db.r6gd.xlarge
```

Managing connections

Multi-AZ DBクラスタは、1つのDBインスタンスではなく、3つのDBインスタンスから構成される

- 各接続は特定のDBインスタンスで処理される
- Multi-AZ DBクラスタに接続する際、指定したホスト名とポート番号は、エンドポイントを表す
- Multi-AZ DBクラスタはエンドポイントを使用してこれらの接続を抽象化
  - すべてのホスト名をハードコードしたり、一部のDBインスタンスが利用できない場合にロードバランシングや接続のリルートのために独自のロジックを書く必要がない

Managing connections

クラスタエンドポイント

- Multi-AZ DBクラスタには、1つのクラスタエンドポイントがある

- Multi-AZ DBクラスタのクラスタエンドポイント(またはライタエンドポイントは)、そのDBクラスタの現在のライタDBインスタンスに接続
  - このエンドポイントは、DDL文やDML文などの書き込み操作を行うことができる唯一のエンドポイント
  - 読み取り操作も可能

- クラスタエンドポイントは、DBクラスタへの読み取り/書き込み接続、フェイルオーバサポートを提供

Managing connections

リーダエンドポイント

- Multi-AZ DBクラスタのリーダエンドポイントは、読み取り可能なスタンバイインスタンスへの読み取り専用接続のロードバランシングをサポート

- 各Multi-AZ DBクラスタには1つのリーダエンドポイント

- リーダエンドポイントは、クエリなどの読み取り操作に使用
  - これらのステートメントをリーダDBインスタンスで処理することで、ライタDBインスタンスのオーバーヘッドを削減

- 同時に実行されるSELECTクエリを処理するために、クラスタの読み取り性能を拡張することが可能

Managing connections

インスタンスエンドポイント

- インスタンスエンドポイントは、Multi-AZ DBクラスタ内の特定の DBインスタンスに接続
- DBクラスタ内の各DBインスタンスは、それぞれ固有のインスタンス・エンドポイントを持つ
- DBクラスタの現在のライタDBインスタンスに対して1つのインスタンスエンドポイント
- DBクラスタ内のリーダDBインスタンスごとに1つのインスタンスエンドポイント
- インスタンスエンドポイントは、DBクラスタへの接続を直接制御したい場合に利用

- 使用例
  - クライアントアプリケーションで、ワークロードの種類に応じてよりきめ細かな負荷分散が必要になる場合があります。このような場合、複数のクライアントがDBクラスタ内の異なるリーダDBインスタンスに接続するように設定することで、読み取りワークロードを分散することが可能

Manage Multi-AZ DB clusters

Rebooting Multi-AZ DB clusters and reader DB instances

DBクラスタの再起動に要する時間は、クラッシュリカバリプロセス、再起動時のデータベースアクティビティ、および特定のDBクラスタの動作に依存

- 再起動時間を短縮するために、再起動時のデータベースのアクティビティをできる限り減らすことを推薦。データベースのアクティビティを減らすことで、転送中のトランザクションのロールバック処理を減らすことが可能

Reboot with a failoverは未対応

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

Determining whether a Multi-AZ DB cluster has failed over

Multi-AZ DBクラスタのフェイルオーバを判断するには

- DBイベントのサブスクリプションを設定し通知する
- Amazon RDSコンソールまたはAPI操作を使用して、DBイベントを表示する
- Amazon RDSコンソール、AWS CLI、およびRDS APIを使用して、Multi-AZ DBクラスタの現在の状態を表示する

Setting the JVM TTL for DNS name lookups

フェイルオーバ機構により、DBインスタンスのドメインネームシステムレコードが自動的に変更され、リーダDBインスタンスに接続するようになる

- DB インスタンスへの既存の接続を再接続する必要がある
- Java仮想マシン環境では、JavaのDNSキャッシュ機構が動作するため、JVMの設定を再設定を要する場合がある

Setting the JVM TTL for DNS name lookups

JVMはDNSの名前検索結果をキャッシュする

- JVMはホスト名を元にIPアドレスに解決する際に、そのIPアドレスをtime-to-liveと呼ばれる特定の期間、キャッシュする

Setting the JVM TTL for DNS name lookups

デフォルトのTTLは、JVMのバージョンやセキュリティマネージャーがインストールされているかどうかによって異なる

- 多くのJVMは、60秒未満のデフォルトTTLを提供
- Oracle のセキュリティ・マネージャーの詳細については、Oracle のドキュメント「The security manager 」を参照

Setting the JVM TTL for DNS name lookups

JVMのTTLを変更するには

- networkaddress.cache.ttl プロパティ値を設定
- JVM を使用するすべてのアプリケーションに対してプロパティ値をグローバルに設定するには、$JAVA_HOME/jre/lib/security/java.security ファイルで networkaddress.cache.ttl を設定
  - networkaddress.cache.ttl=60

- プロパティを特定のアプリケーションにのみ設定するには、ネットワーク接続が確立される前に、アプリケーションの初期化コードでnetworkaddress.cache.ttlを設定
  - java.security.Security.setProperty("networkaddress.cache.ttl" , "60")

Amazon Aurora and RDS Multi-AZ DB cluster deployments

- Amazon Aurora MySQLおよびPostgreSQL互換のリレーショナルデータベース
  - Auroraは、従来のエンタープライズデータベースのパフォーマンスと可用性を、オープンソースデータベースのシンプルさとコスト効率のを組み合わせている
- 高い読み込み性能が必要な場合、AuroraはAuroraクラスタごとに最大15のリードレプリカをサポート
- 数千の同時セッションを処理し、AWSリージョン間を含めて毎秒数十万のトランザクションを必要とするデータベースを必要とする場合には、Auroraの利用を推薦
- Amazon Auroraは、他にもパフォーマンス、可用性、管理性の向上など独自機能を提供
  - サーバーレス運用、自動でスケールするリードレプリカ、ディザスタリカバリ用のグローバルデータベース、論理エラーを瞬時にオンラインで修正するバックトラック機能、S3への自動連続増分バックアップ、高速なデータベースクローニングなど

Monitor a Multi-AZ DB cluster

Monitor a Multi-AZ DB cluster

Amazon RDSでは、Multi-AZ DBクラスタ用のCloudWatchメトリクスを追加

- ReplicationLag
- WriteIOPSLocalStorage
- WriteLatencyLocalStorage

Summary

Summary

- Amazon Auroraだけでなく Amazon RDSも機能改善を行っている
  - アプリケーションの用途に応じてデータベースを選択
- Amazon Aurora Serverless v2を利用しデーターベースインスタンスのキャパシティ管理を自動化
  - 例: Shardごとにクラスタをアサイン
  - Mixedコンフィグレーションでを利用することで定常的に利用するキャパシティと予測が難しいワークロードの共存やコスト最適化が可能
- アップグレードや新機能の利用前にはテストを推薦
  - 実ワークロードでの検証を推薦
  - Previewは本番環境での利用は禁止
  - 検証結果のフィードバックなどお待ちしています

Thank you!