Automated Backups - Investigate PITR Backup Billing Charges

Last updated by | Keith Elmore | Apr 5, 2021 at 7:56 AM PDT

Contents

Issue

vCore databases are billed separately for the storage consumed by the backup files. Since enabling this functionality in June 2019, we have received many IcMs and cases that inquire and/or express dissatisfaction about the charges.

Most customers wonder why they are suddenly being charged for this resource when they previously had automated SQL DB backup functionality without being charged in the past. Further, they'll then ask why the billed amount is the quantity that it is. This TSG details on how to investigate this usage as well as provides boilerplate responses and other helpful resources.

Billing Model

The billing meter is called "Single/Elastic Pool PITR Backup Storage" which charges at a rate of $/GB/month and provides hour-granular billing.

https://azure.microsoft.com/en-us/pricing/details/sql-database/single/

https://docs.microsoft.com/en-us/azure/sql-database/sql-database-automated-backups#storage-costs

Standalone Databases

We aggregate the size of all in-retention backups and deduct the provisioned database max size. Another way of phrasing this is we give a 1x database size discount on backup storage. For example, if a 32 GB database has 64 GB of backup storage, they will be charged for 32 GB of backup storage.

This charge will be identified by resourceUri of the database, e.g:

/subscriptions/Input SubscriptionId here/resourceGroups/work/providers/Microsoft.Sql/servers/adeal-stage-neu/databases/gp

Pooled Databases

We aggregate the size of all in-retention backups for each database in the pool. We then aggregate the consumption for all databases in the pool and deduct the provision pool storage limit. For example, if a 32 GB resource pool has 2 databases that both have 64 GB of backup storage usage, customers will be charged for 96 GB of backup storage.

This charge will be identified by resourceUri of the resource pool, e.g:

/subscriptions/Input SubscriptionId here/resourceGroups/work/providers/Microsoft.Sql/servers/adeal-stage-neu/elasticPools/pool

Dropped Databases (standalone and pooled)

We bill for backups even after the database is dropped. This is because we allow customers to restore dropped databases and must retain backups to do so. Once dropped, the backup charges will slowly decrease until days_since_dropped > backup_retention_days.

This charge will be identified by resourceUri of the restorableDroppedDatabase, e.g

/subscriptions/Input SubscriptionId here/resourceGroups/work/providers/Microsoft.Sql/servers/adeal-stage-neu/restorableDroppedDatabases/gp,132063075017230000

Common Questions

Question	Response
Why do I not see charges for all my databases?	We bill for backup storage for VCore databases; DTU databases do not have a separate charge for this as it is included in the base price of the database. Additionally, depending on database usage, the storage used for backups may not exceed the provisioned database/pool max size, in which case there would be no charge.
Why is this charge suddenly showing up for my databases?	The sudden occurrence of this charge is expected; this billing functionality was just enabled in June 2019
Why was I not notified about the new charge?	Communications about this new billing meter were sent back in December 2018, however, they could have been easily missed. We apologize if this was the case. An example communication email can be found on sharepoint, however, please do not forward this to customers since this contains customer information. Feel free to forward snippets from the body of the email, if applicable.
How can I reduce this charge?	Currently, the best way to reduce the backup storage consumption is by reducing the backup retention setting for the database. The current minimum retention allowed is 1 day. In the future, customers will be able to select their backup account storage type and change it from RA-GRS (ReadAccessible - GlobalyRedundantStorage) to LRS (Locally redundant Storage) which costs less since it would not support Geo-Restore.
I have no use for automated backup or point-in-time restore. Can I disable backups?	Response for the customer is No. Note for Engineer : Automated backups not only protect the customer, but protect Microsoft's ability to mitigate the off-chance of a database or storage corruption.
I reduced the retention on a database and my bill did not decrease.	It can take up to 24 hours for a change of retention to impact the billed usage.
Why is my bill for backup storage so large?	The backup size ultimately depends on database workload. Please see Investigate Billed Quantity to get potential insights for this.

Expose backup storage metrics to customer	This is work planned that we expect to be available in prod by first quarter 2020.

How backup storage pile up

SQL Database supports self-service for point-in-time restore (PITR) by automatically creating full backup, differential backups, and transaction log backups. Full database backups are created weekly, differential database backups are generally created every 12 hours, and transaction log backups are generally created every 5 - 10 minutes, with the frequency based on the compute size and amount of database activity. The first full backup is scheduled immediately after a database is created. It usually completes within 30 minutes, but it can take longer when the database is of a significant size.

After the first full backup, all further backups are scheduled automatically and managed silently in the background. The exact timing of all database backups is determined by the SQL Database service as it balances the overall system workload. You cannot change or disable the backup jobs.

In order to support restore to any point in time within the retention period, we retain the entire "backup chains" (full + differentials + log backups) between each full backup until they are no longer needed. This means that we are likely storing multiple "backup chains", based on the configured retention period for the database.

How do we calculate the value:

The billing meter for PITR (Single/Elastic Pool PITR Backup Storage) charges at a rate of $/GB/month and provides hour-granular billing.

"hour-granular billing": this means that we, every hour, check the backup storage consumption and communicate it to the billing team.

Since we charge "$/GB/month", the excess value we communicate has to be calculated from a monthly value to an hourly value.

Example:

- At 10 am today the backup storage usage for the database/pool was 750GB
- The reserved storage, at 10 am today, for the database/pool was 500GB

This means that we, at 10am today, would need to charge for 250GB excess usage (since 1x of the instance storage size is free)

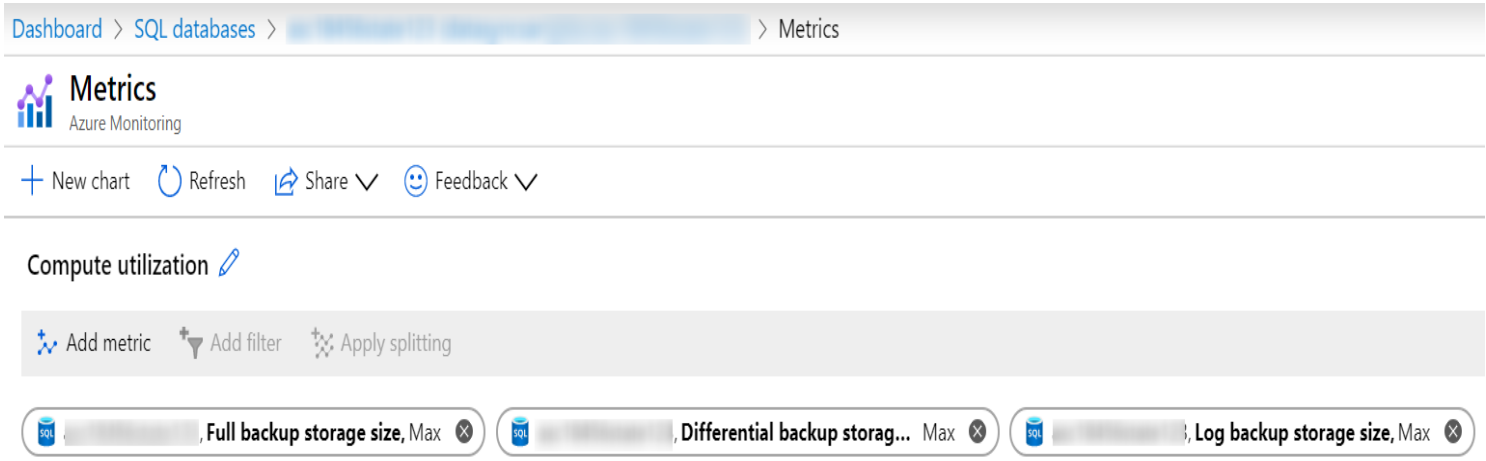
So, we charge 250GB/31 days/24h = 0,336 GB to transform a monthly charge to an hourly charge.

Recommended actions for the customer

Review the backup retention period of your Azure SQL Databases, consider decreasing the retention period to the lowest possible value to use less storage and optimize your costs.
Learn how to monitor backup storage costs in Azure Cost Management.

You can also see metrics for backup consumption in Metrics blade, you will have metrics for:

- Full backup storage size
- Differential backup storage size
- Log backup storage size



You also might want to consider additional actions to reduce backup costs for your databases:

- Avoid performing large write operations more frequently than needed, such as index rebuilds.
- For large data load operations consider using clustered columnstore indexes, reduce number of non-clustered indexes, and execute bulk load operations with row count around 1M.
- In General Purpose service tier, the provisioned storage for data is about 50% less cost than the backup storage. Customers with continuous backup storage excess charges might consider increasing the size of the data storage to save on the backup storage.
- Use TempDB in your ETL logic for storing temporary results, instead of storing in permanent tables.
- Consider turning off TDE encryption for that databases that do not contain sensitive data (development databases, for instance). Backups for the non-encrypted databases are typically compressed with a higher compression ratio, hence using less backup storage.

Investigate Billed Quantity

The following query will report the usage that gets sent to Microsoft commerce team (responsible for translating SQL DB's reported usage to an actual monetary bill)

Execute: [Web] [Desktop] [Web (Lens)] [Desktop (SAW)] https://sqlstage.kustomfa.windows.net/sqlazure1

```
let HoursPerMonth = 744;
PostReportedUsage2
| where InfoFieldMeteredServiceType == "SQL Database RAGRS Backup Storage"
| where ServerName =~ "{server_name}" or ElasticPoolName =~ "{pool_name}" or DatabaseName =~ "
{database_name}"
| project EventDateTime, ResourceUri, ServerName, ElasticPoolName, DatabaseName, ElasticPoolId, DatabaseId,
UsageResourceQuantity, CurrentCumulativeBackupConsumptionInGB =
UsageResourceQuantity*HoursPerMonth
```

```
//Hourly consumption reported, should match billing details
PostReportedUsage2
| where EventDateTime >= datetime({StartTime}) and EventDateTime <= datetime({EndTime})
| where ServerName =~ '{ServerName}'
| where InfoFieldMeteredServiceType == "SQL Database RAGRS Backup Storage"
| extend RestorableDroppedDB = iif(ResourceUri contains "restorableDroppedDatabase", substring(ResourceUri,
indexof(ResourceUri, "restorableDroppedDatabase")+27),'')
| summarize sum(UsageResourceQuantity), count(UsageResourceQuantity) by tolower(ResourceUri),
ServerName, ElasticPoolName, DatabaseName, RestorableDroppedDB, bin(EventDateTime, 1d)
| project-rename BackupConsumptionInGB = sum_UsageResourceQuantity, NumberOfHourlyRecords =
count_UsageResourceQuantity, Date = EventDateTime
| project Date, ServerName, DatabaseName, ElasticPoolName, RestorableDroppedDB, NumberOfHourlyRecords,
round(BackupConsumptionInGB,4)
| order by Date asc
```

To get insight into the charges, execute the following:

Execute: [Web] [Desktop] [Web (Lens)] [Desktop (SAW)] https://sqlazureneu2.kustomfa.windows.net/sqlazure1

```
MonManagement
| where event =~ 'sqldb_backup_billing_complete' and logical_database_id =~ '{LogicalDatabaseId}'
| project billingCalculationTimestamp=originalEventTimestamp, logical_database_id,
full_backup_cumulative_size_mb=full_backup_cumulative_size,
diff_backup_cumulative_size_mb=diff_backup_cumulative_size,
log_backup_cumulative_size_mb=log_backup_cumulative_size,
cumulative_backup_storage_size_mb=todouble(backup_storage_size)
| extend full_ratio = full_backup_cumulative_size_mb/cumulative_backup_storage_size_mb,
diff_ratio=diff_backup_cumulative_size_mb/cumulative_backup_storage_size_mb,
log_ratio=log_backup_cumulative_size_mb/cumulative_backup_storage_size_mb
| join kind=inner
(
    MonAnalyticsDBSnapshot
    | where logical_database_id =~ '{LogicalDatabaseId}'
    | project logicalDbSnapshotTimestamp=TIMESTAMP, logical_server_name, logical_database_name,
logical_database_id=tolower(logical_database_id), logical_resource_pool_id,
    logical_database_max_size_mb=max_size_bytes/1024.0/1024.0, backup_retention_days,
sql_instance_name=tolower(sql_instance_name), physical_database_id=tolower(physical_database_id)
)
on logical_database_id
| where billingCalculationTimestamp > logicalDbSnapshotTimestamp
| summarize arg_max(logicalDbSnapshotTimestamp, *) by billingCalculationTimestamp, logical_database_id
| join kind=inner
(
    MonDmIoVirtualFileStats
    | where logical_database_id =~ '{LogicalDatabaseId}'
    | where type_desc == 'ROWS' and is_primary_replica == 1
    | summarize physical_database_size_on_disk_mb=sum(size_on_disk_bytes)/1024.0/1024.0 by TIMESTAMP,
logical_database_id
    | project physicalDbSnapshotTimestamp=TIMESTAMP, physical_database_size_on_disk_mb,
```

```
logical_database_id=tolower(logical_database_id)
) on logical_database_id
| where billingCalculationTimestamp > physicalDbSnapshotTimestamp
| summarize arg_max(physicalDbSnapshotTimestamp, *) by billingCalculationTimestamp, logical_database_id
| project-away logical_database_id1, logical_database_id2, sql_instance_name, physical_database_id
| project billingCalculationTimestamp, logical_server_name, logical_database_name,
full_backup_cumulative_size_mb, diff_backup_cumulative_size_mb,
log_backup_cumulative_size_mb, cumulative_backup_storage_size_mb, logical_database_max_size_mb,
backup_retention_days
| project-rename BillingCalculationTimestamp=billingCalculationTimestamp, ServerName=logical_server_name,
DatabaseName = logical_database_name,
FullBackupCumulativeSizeMB=full_backup_cumulative_size_mb,
DiffBackupCumulativeSizeMB=diff_backup_cumulative_size_mb,
LogBackupCumulativeSizeMB=log_backup_cumulative_size_mb,
CumulativeBackupStorageMB=cumulative_backup_storage_size_mb,
DatabaseMaxSizeMB=logical_database_max_size_mb, BackupRetentionDays=backup_retention_days
| order by BillingCalculationTimestamp asc

//(from Matteo Teruzzi) Alternative query version by server name
MonAnalyticsDBSnapshot
| where logical_server_name =~ '{server_name}'
| project logicalDbSnapshotTimestamp=TIMESTAMP, logical_server_name, logical_database_name,
logical_database_id=tolower(logical_database_id), logical_resource_pool_id,
logical_database_max_size_mb=max_size_bytes/1024.0/1024.0, backup_retention_days,
sql_instance_name=tolower(sql_instance_name), physical_database_id=tolower(physical_database_id)
| join (
    MonManagement
    | where event =~ "sqldb_backup_billing_complete"
    | project billingCalculationTimestamp=originalEventTimestamp, logical_database_id,
full_backup_cumulative_size, diff_backup_cumulative_size, log_backup_cumulative_size, backup_storage_size
) on logical_database_id
| where billingCalculationTimestamp > logicalDbSnapshotTimestamp
| project billingCalculationTimestamp, logicalDbSnapshotTimestamp, logical_database_id,
logical_database_name, full_backup_cumulative_size_mb=full_backup_cumulative_size,
diff_backup_cumulative_size_mb=diff_backup_cumulative_size,
log_backup_cumulative_size_mb=log_backup_cumulative_size,
cumulative_backup_storage_size_mb=todouble(backup_storage_size), sql_instance_name, physical_database_id
| extend full_ratio = full_backup_cumulative_size_mb/cumulative_backup_storage_size_mb,
diff_ratio=diff_backup_cumulative_size_mb/cumulative_backup_storage_size_mb,
log_ratio=log_backup_cumulative_size_mb/cumulative_backup_storage_size_mb
| summarize arg_max(logicalDbSnapshotTimestamp, *) by billingCalculationTimestamp, logical_database_id,
logical_database_name
| join kind=inner
(
    MonDmIoVirtualFileStats
    | where type_desc == "ROWS" and is_primary_replica == 1
    | summarize physical_database_size_on_disk_mb=sum(size_on_disk_bytes)/1024.0/1024.0 by TIMESTAMP,
logical_database_id
    | project physicalDbSnapshotTimestamp=TIMESTAMP, physical_database_size_on_disk_mb,
```

```
logical_database_id=tolower(logical_database_id)
) on logical_database_id
| where billingCalculationTimestamp > physicalDbSnapshotTimestamp
| summarize arg_max(physicalDbSnapshotTimestamp, *) by billingCalculationTimestamp
| project-away logical_database_id1, logical_database_id, sql_instance_name, physical_database_id

//Simple version based on Matteo query - Sergio Fonseca

let ServerName = "Server";
let TimeRange = ago(50d);
MonAnalyticsDBSnapshot
| where TIMESTAMP >= TimeRange
| where logical_server_name =~ ServerName
| distinct logical_database_id, logical_database_name
| join kind=leftouter (
   MonManagement
   | where TIMESTAMP >= TimeRange
   | where event =~ "sqldb_backup_billing_complete"
   | project billingCalculationTimestamp=originalEventTimestamp, logical_database_id =
toupper(logical_database_id),
            full_backup_cumulative_size, diff_backup_cumulative_size, log_backup_cumulative_size,
backup_storage_size
) on logical_database_id
| order by logical_database_name asc, billingCalculationTimestamp asc
| project logical_database_name, billingCalculationTimestamp, full_backup_cumulative_size,
        diff_backup_cumulative_size, log_backup_cumulative_size, backup_storage_size
```

This will separate the billed backup storage by backup type and calculate the ratio that each type contributed to the total. It will also list the backup retention setting that was present when the billing calculation was performed, as well as the logical database max size and physical database size on disk.

Usually, databases fall into two categories:

Majority of backup storage from DIFF + LOG backups

This typically indicates that the database undergoes a large amount of data churn. Giving this explanation is usually sufficient for customers.

Majority of backup storage from FULL backups

This indicates that the data within the database is somewhat stable. Look at the physical database size returned above to get an estimate on the size of the database, which will explain the large FULL backup sizes.

Example

IcM: https://portal.microsofticm.com/imp/v3/incidents/details/135558403/home

The standard response I have provided is:

The sudden occurrence of this charge is expected; this billing functionality was just enabled in June. Comms were sent back in December 2018, however, they could have been easily missed by the customer. Additionally,

this billing functionality has a slow-start mechanism when it was enabled, so it is expected that the charges associated with PITR Backup Storage have slowly increased since they started appearing in June.

I took a look at the databases, however, to inspect the backups being billed for. It is important to note that the customer previously had 35 day retention configured for the databases, and it was only updated to 7 days on 2019-07-25. The bill has not refreshed since they performed that update, but, I expect the bill to be stable going forward.

For database <database_name>:

full_backup_cumulative_size	diff_backup_cumulative_size	log_backup_cumulative_size	backup_stora
<full_size>	<diff_size>	<log_size>	<tot

We see that a majority of backup consumption is from LOG + DIFF backups. This indicates that the database has a large amount of data churn.

We are rolling out a new database metric that will expose the cumulative backup size, broken down by FULL/DIFF/LOG.

https://feedback.azure.com/forums/217321-sql-database/suggestions/38130346-ability-to-view-backup-storage-usage-via-portal-p

Note

SQL DB shipped a regression end of July that effectively disabled PITR billing for all databases. We rolled out a fix in Aug and it made it WW by beginning of Sep. However this meter has a slow-start mechanism so takes a few weeks to accumulate the complete backup usage after it gets turned back on.

Outside of that, there has been no change to the way we charge for PITR backups.

tags: pitr billing, backup billing, SQLDB_RAGRSBackupStorage, Single/Elastic Pool PITR Backup Storage

How good have you found this content?