

# PITR Backup Billing Charges in MI

Last updated by | Charlene Wang | Jun 18, 2021 at 2:27 AM PDT

## Contents


- Public Doc Reference
  - Automated backups
  - Backup storage consumption on Managed Instance explai...
  - Configuring backup storage redundancy in Azure SQL
  - Fine tuning backup storage costs on SQL Managed Instance
- Summary (public)
- Billing Model (internal)
  - How backup storage piles up (public):
  - How do we calculate the value (public):
- Communications
  - Communication sent to customers (on Nov 26th) that got ...
  - Review the backup retention period for your Azure SQL Da...
- Actions for CSS (internal)
  - For all customers who are at risk of leaving SQL DB MI plat...
  - If customer doesn't have an EA
  - If customer has EA (Enterprise Agreement)
  - In addition, please ask the impacted customer the below q...
- Recommended actions for the customer (public, this sectio...
  - (new) Additional configuration options for the customers f...
- Configuring backup storage redundancy in Azure SQL
- Investigate Billed Quantity using ASC
- Check current database retention period using CMS
- Known issue: Bug in the backup storage billing system con...
  - Investigate backup chains and database operations
  - RCA
- Investigate Billed Quantity

## Public Doc Reference


### Automated backups

<https://docs.microsoft.com/en-us/azure/sql-database/sql-database-automated-backups?tabs=single-database>

### Backup storage consumption on Managed Instance explained

<https://techcommunity.microsoft.com/t5/azure-sql/backup-storage-consumption-on-managed-instance-explained/ba-p/1390923> 

## Configuring backup storage redundancy in Azure SQL

<https://techcommunity.microsoft.com/t5/azure-sql/configuring-backup-storage-redundancy-in-azure-sql/ba-p/1554322> 

## Fine tuning backup storage costs on SQL Managed Instance

<https://techcommunity.microsoft.com/t5/azure-sql/fine-tuning-backup-storage-costs-on-sql-managed-instance/ba-p/1390935> 

Note, please only share the information that is stated as public

## Summary (public)

The promotional period with free unlimited **backup storage for Azure SQL Database managed instance** was in effect from March 7<sup>th</sup> 2018 until November 1<sup>st</sup> 2019.

On November 1<sup>st</sup> this year we started [charging for the excess backup storage](#).

As a result, customers will be charged for backup storage that exceeds the amount of reserved storage space specified for managed instance during initial deployment or after storage scaling.

We have already informed customers about this change 30 days prior the rollout in October 2019 with details of the change and how to monitor charges announced through the public [Azure service announcement](#), and through sending personalized emails to subscription owners.

## Billing Model (internal)

The billing meter is called "**SQL Database Managed Instance PITR Backup Storage**" which charges at a rate of \$/GB/month and provides hour-granular billing.

We are adapting the same model as single/pooled SQL DB - customers will get the equal amount of free backup storage space as the data storage space purchased, regardless of the backup retention period set. Exceeding the use of backup storage above the free space will result in costs of about \$0.20 - \$0.24 per GB/month in US regions, or see the [pricing page](#) for details for your region.

## Example

For example, if customer purchases MI with 4TB of data storage, they will get 4TB of backup storage for free, regardless of the DB backup retention period set. The provided backup storage space is used for all databases and all backups cumulatively. Exceeding usage over the free space in this example will come at surcharge. This means that customers with 7 days backup retention on their databases will most likely not exceed usage of the free backup space, whereas customers with 35 days backup retention period are very likely to exceed the usage of the free backup space (\*this is all relative and dependent on customers' actual database sizes).

## How backup storage piles up (public):

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 (public):

The billing meter for PITR (**SQL Database Managed Instance 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 instance was 750GB
- The reserved storage, at 10 am today, for the database 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  $250\text{GB} / 31 \text{ days} / 24\text{h} = 0,336 \text{ GB}$  to transform a monthly charge to an hourly charge.

We do these steps every hour.

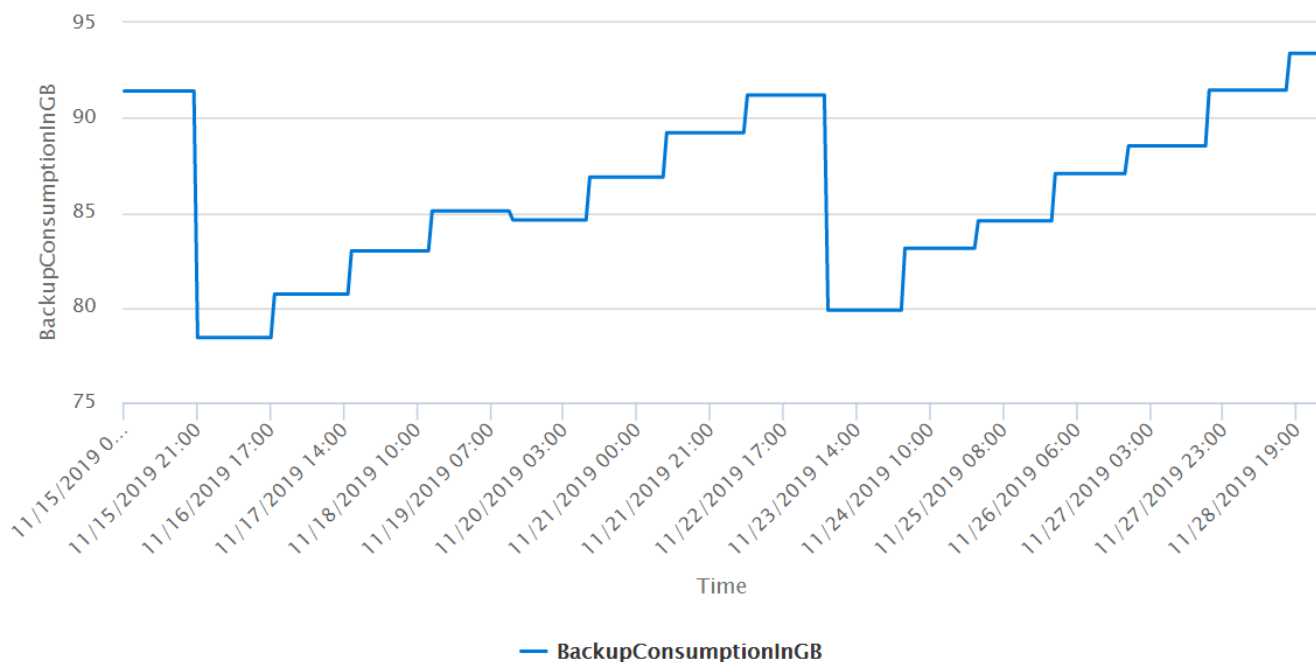
### (Internal only)

We actually don't check the storage usage every hour, we do it in bigger intervals, that's why you see a ladder in the billing metric, where every step has the same value for hours.

Telemetry is exhausted several hours before billable value changes. This means that customer can see different values for backup size in the query provided and on the bill for the same timestamp (but the new value will become the billable one soon).

## PITR Backup Storage | SQL Database Managed Instance PITR Backup Storage billing meter

Kusto Query ☺ ☹



For single databases, the consumption is easier to see. For elastic pools and managed instance, since each database has its own time for full/diff/log backups, it's harder to check where the main consumption is coming from (which database, which kind of backup, must like a combination of both).

## Communications

### Email that went out to customers



## Read about upcoming billing changes to Azure services

You're receiving this email because your organization uses one or more Azure services that will soon experience a billing change, and we're committed to communicating all changes in advance of your affected invoice.

### Azure SQL Database

Effective November 1, 2019, we'll begin charging for backup storage on Azure SQL Database managed instance. Complete details about the [new pricing](#) will be available on that date.

## Azure SQL Database—Billing for managed instance backup storage is now in effect

<https://azure.microsoft.com/en-in/updates/billing-for-managed-instance-backup-storage-starts-november-1/>

### And pricing page

<https://azure.microsoft.com/en-us/pricing/details/sql-database/managed/>

**Communication sent to customers (on Nov 26th) that got more than 20% increase in the bill due to backup storage consumption**

### Review the backup retention period for your Azure SQL Database managed instances

*You're receiving this email because you use Azure SQL Database managed instances.*

On November 1, 2019, we started charging for [backup storage](#) for Azure SQL Database managed instances (see [pricing details](#)). As a result, you'll be charged for backup storage that exceeds the amount provided when you provision data storage.

To ensure you're not charged for backup storage you may not need, review the backup retention period of your managed instances. A shorter retention period uses less backup storage and will be less likely to incur charges.

#### Recommended action

Review the backup retention period of your Azure SQL Database managed instances. If you've extended it beyond the default of seven days, consider [decreasing the retention](#) period to use less storage and optimize your costs.

Learn how to [monitor backup storage costs](#) in Azure Cost Management.

If you have questions, please [contact us](#).

## Actions for CSS (internal)

**For all customers who are at risk of leaving SQL DB MI platform and have experienced more than 100% (1x) of increase in the bill**

If the customer has experienced more than 100% of billing increase, reach out to Vitor Tomaz to have a look at the case (he will evaluate customer's risk of leaving the platform) and approve the following procedure. Upon Vitor's approval, execute the following:

### If customer doesn't have an EA

Raise a collaboration task with the billing team, using the same justification you can find for EA customers below.

## If customer has EA (Enterprise Agreement)

Create the EA billing refund request ticket using the below details.

**Do not share with customers the fact that you are issuing a refund request on customers behalf until you actually get the refund request approved.** We need to avoid DSAT in case the refund is not approved for some reason.

Keep Dani Ljepava (danil) and Vitor Tomaz (vitomaz) in copied in the refund request communications.

Help the customer in working with them to take advantage of clustered columnstore indexes, and reduction of non-clustered indexes for their data workloads IF their workload is analytical, data mart \ data warehouse (this type of workload is the most likely to generate excessive backup storage consumption).

Here is the information needed to make the EA billing refund:

- Submit refund request by going to Azure EA support portal: <https://support.microsoft.com/en-us/supportrequestform/e114582c-4e51-af46-10b1-1f0cc141e133>. Fill out the information as suggested below:
- Issue title: Refund request for Azure SQL MI PITR storage
- Name, email and phone: provide your own and also Dani Ljepava (danil) contact details (you will act as customer's advocate)
- Issue category: Billing and invoicing
- Customer name: (fill)
- EA enrollment number: (fill)
- Billing issue: Refund request
- Details of the issue:
  - "This customer is EA that has been severely impacted by the sudden increase in charges from a new billing meter "mi pitr backup storage" starting from November 1<sup>st</sup>, 2019 up to date. While the billed usage appears to be correct, the credit would be a good faith measure to help remedy the suddenness of the charges as well as give them additional time to reduce the backup sizes from their side. I've spoken with them directly and they are open to making changes, but still frustrated by the cost. We've communicated that the billed amount was correct and set the expectations about these charges going forward. Additionally, product group did not expose sufficient tooling to customers to understand the scale of these new charges and have the ability to fine tune their backup storage consumption. These additional tools that will help customer further fine tune its consumption are expected to be rolled out in Q1 2020. Due to this I believe it is best we start the remediation process with a refund of the backup storage charges starting from the date of introducing the sudden billing change November 1<sup>st</sup>, 2019, up to date."

**In addition, please ask the impacted customer the below questions and then respond back to the PG:**

INTERNAL:



We can manually help in some ways even before customer-facing tools are introduced in reducing diff backup time, and setting the active DB retention period, but this needs to be evaluated on case by cases only IF it makes significant reduction in the bill for a particular customer.

Please note that we are only willing to take this into consideration in case the invoice was dramatically increased (more than 200%). Please take this into consideration that it cannot be offered to all other customers whose impact of the billing charge is much less, they would need to wait for the official tools to become available (Q1 2020).




Please ask the following questions and send the replies to: mdcsqlmipm, Dani Ljepava (danil), and Vitor Tomaz (vitomaz):

- Q: Thus far did you do a restore of your databases in production (e.g. what is your need for restoring prod databases like)?
- Q: In case you need to restore your largest database, please let us know what is the acceptable longest restore time for such database? We are asking this in the context of further reducing the frequency of diff backups which might consequently considerably increase the restore time, but will reduce the billing charges. Please bear in mind while evaluating this that the average restore speed from the full backup is 360 GB / hour, whereas for the log backups the average restore speed is 60-70 GB / hour. Based on your response on what is the acceptable longest restore time for your database, we can take into consideration to manually implement this change for you if it would further provide significant savings. Please note that if turns out to be the case, we will need your explicit written consent that you understand the consequences and agree with making such changes.
- Q: In case that we can deliver 1 day backup retention period sooner (to be determined if/when this would be possible), please let us know if this is something that would be acceptable for you? In other words, would you be comfortable of being able to go back only 1 day in terms of the backup? This will reduce the billing cost (we can also provide an estimate), however the consequence is that your time-window for restore from a data corruption would be minimal. If this is something you'd feel comfortable with, you would need to provide us with an explicit written consent for making such changes before the tooling to do this yourselves becomes available. We will evaluate each request on case by case basis.

## Recommended actions for the customer (public, this section can/should be shared with customers)

Review the backup retention period of your Azure SQL Database managed instances. If you've extended it beyond the default of 7 days, consider [decreasing the retention period](#)  to the lowest possible value (1 day) to use less storage and optimize your costs. We now support between 1 and 35 backup retention in days for databases. Learn how to [monitor backup storage costs](#)  in Azure Cost Management.

You also might want to consider reducing the backup storage retention for dropped databases to 1 day or remove them entirely (0 days) using the following PowerShell script:



- To get a list of dropped databases: <https://docs.microsoft.com/en-us/powershell/module/az.sql/get-azsqldeletedinstancedatabasebackup?view=azps-3.1.0> 
- To get retention period policies set for the databases: <https://docs.microsoft.com/en-us/powershell/module/az.sql/get-azsqlinstancedatabasebackupshorttermretentionpolicy?view=azps-3.1.0> 
- To set a new retention period: <https://docs.microsoft.com/en-us/powershell/module/az.sql/set-azsqlinstancedatabasebackupshorttermretentionpolicy?view=azps-3.1.0> 

## (new) Additional configuration options for the customers for a better control of backup costs



**We now support between 1 and 35 backup retention in days for databases.**

**We also support between 0 (no retention) and 1 backup retention for deleted databases.**

Minimum required version of Az.SQL module needs to be v2.6.0 - [update here](#) , or execute through Azure [CloudShell](#) .


Here's some examples:

```
#Get current configuration of a database
$resourceGroup = ''
$instance = ''
$database = ''
$days = 1
Get-AzSqlInstanceDatabaseBackupShortTermRetentionPolicy -ResourceGroupName $resourceGroup -InstanceName $insta

#Set a new retention on a database (valid backup retention in days must be between 1 and 35)
$resourceGroup = ''
$instance = ''
$database = ''
$days = 1
Set-AzSqlInstanceDatabaseBackupShortTermRetentionPolicy -ResourceGroupName $resourceGroup -InstanceName $insta

# Set PITR Retention on all Databases
# Valid backup retention in days must be between 1 and 35
$resourceGroup = ''
$instance = ''
$days = 1
Get-AzSqlInstanceDatabase -ResourceGroupName $resourceGroup -InstanceName $instance | Set-AzSqlInstanceDatabas

# Set PITR Retention on all Deleted Database Backups
# Valid backup retention in days must be between 0 (no retention) and 1
$resourceGroup = ''
$instance = ''
$days = 0
Get-AzSqlDeletedInstanceDatabaseBackup -ResourceGroupName $resourceGroup -InstanceName $instance | Set-AzSqlIn
```

Fine tune the backup storage consumption. The excess backup storage consumption will depend on the workload and size of the individual databases. You can consider implementing some of the tuning techniques to further reduce your backup storage consumption you can find at <https://docs.microsoft.com/en-us/azure/sql-database/sql-database-automated-backups?tabs=single-database#>  fine-tune-the-backup-storage-consumption

## Configuring backup storage redundancy in Azure SQL

See more at <https://techcommunity.microsoft.com/t5/azure-sql/configuring-backup-storage-redundancy-in-azure-sql/ba-p/1554322> 

## Investigate Billed Quantity using ASC

Telemetry is exhausted several hours before billable value changes. This means that customer can see different values for backup size in the query provided and on the bill for the same timestamp (but the new value will become the billable one soon).



In Instant Content of SQL Managed Instance Troubleshooter you can find:

Summary

Availability

Connectivity

Networking

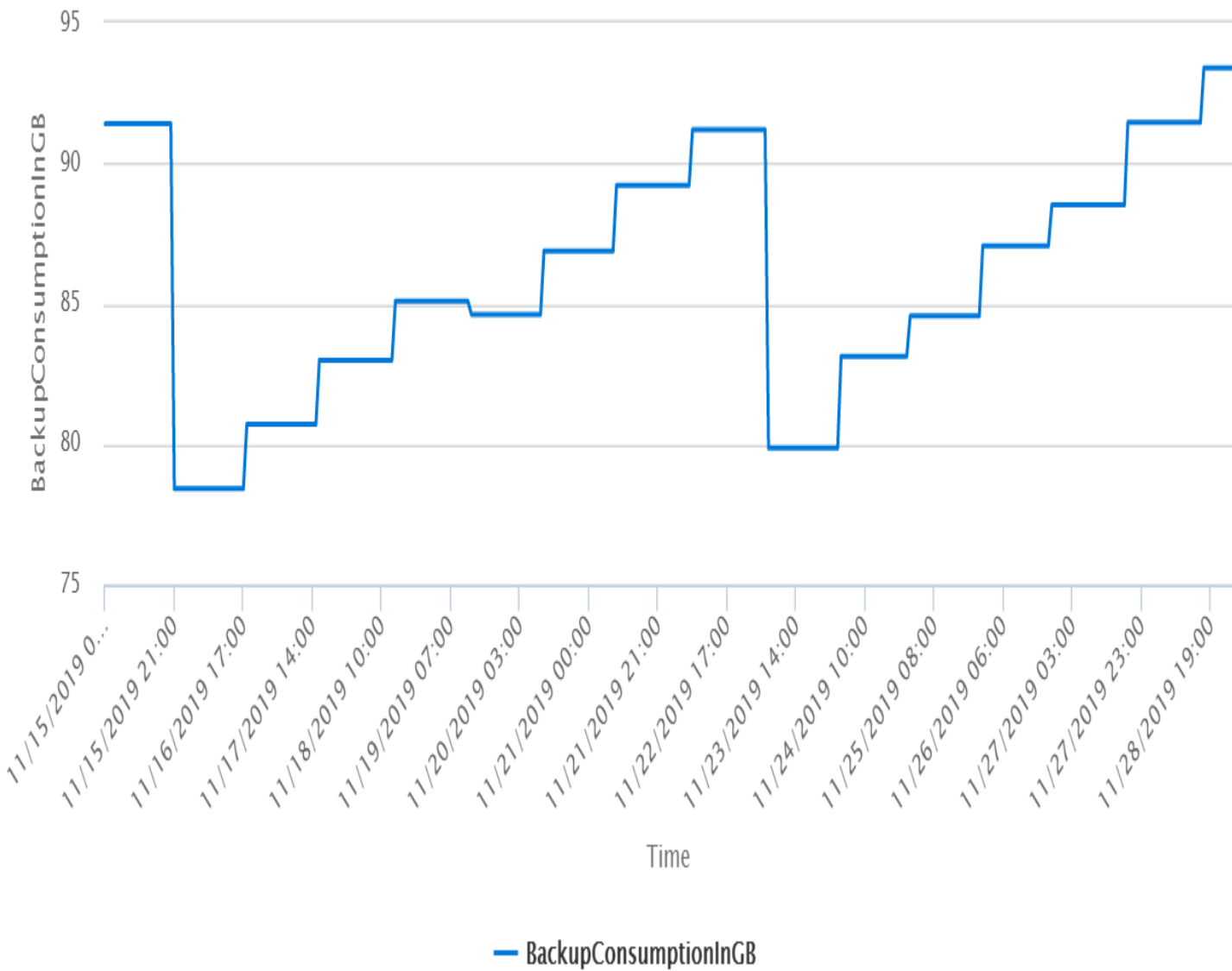
Provisioning

Performance

Instant Content

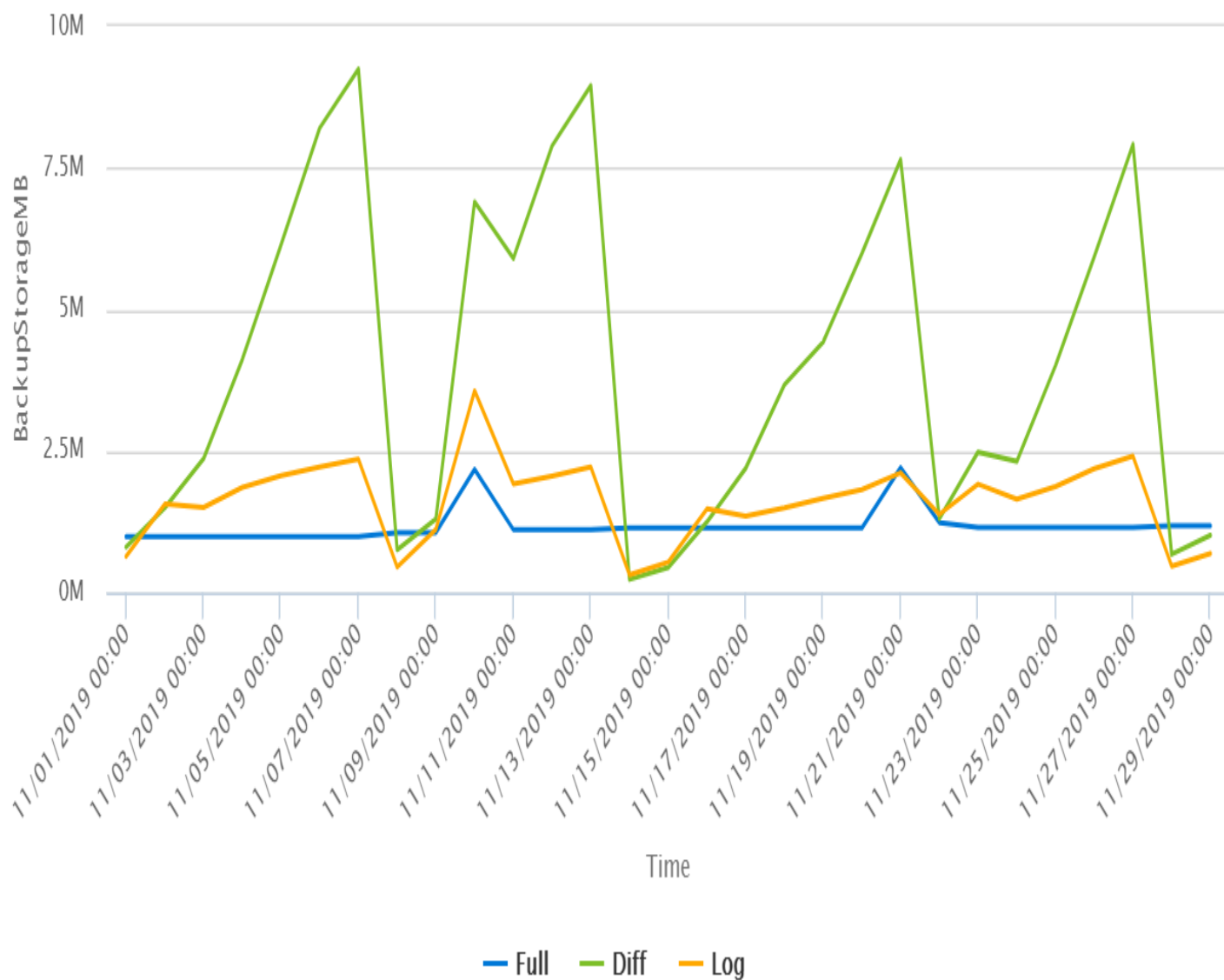
PITR Backup Storage | SQL Database Managed Instance PITR Backup Storage billing meter

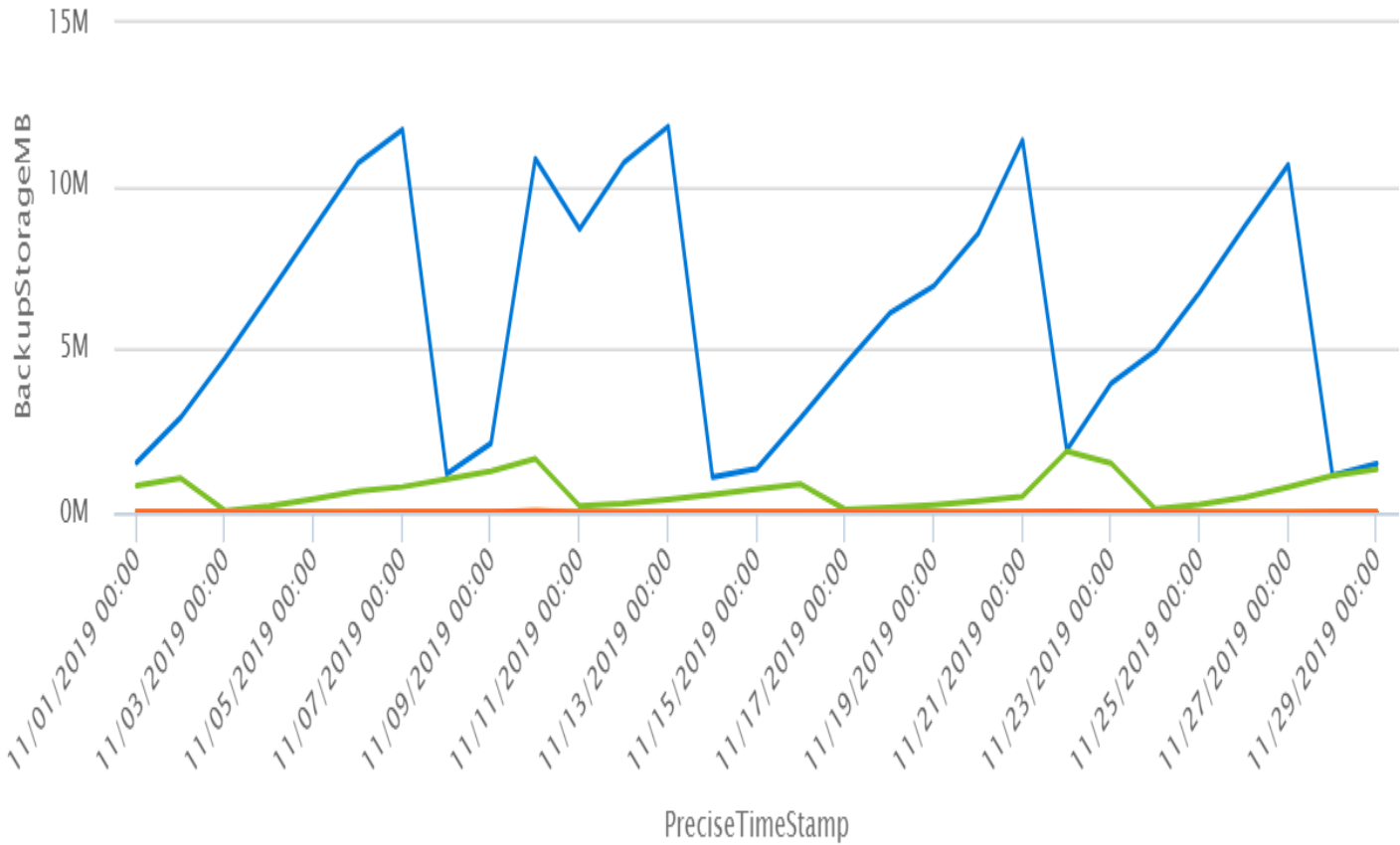
Kusto Query ☺ ☹



## PITR Backup Storage | Daily accumulated storage consumption per backup type

Kusto Query ☺ ☹





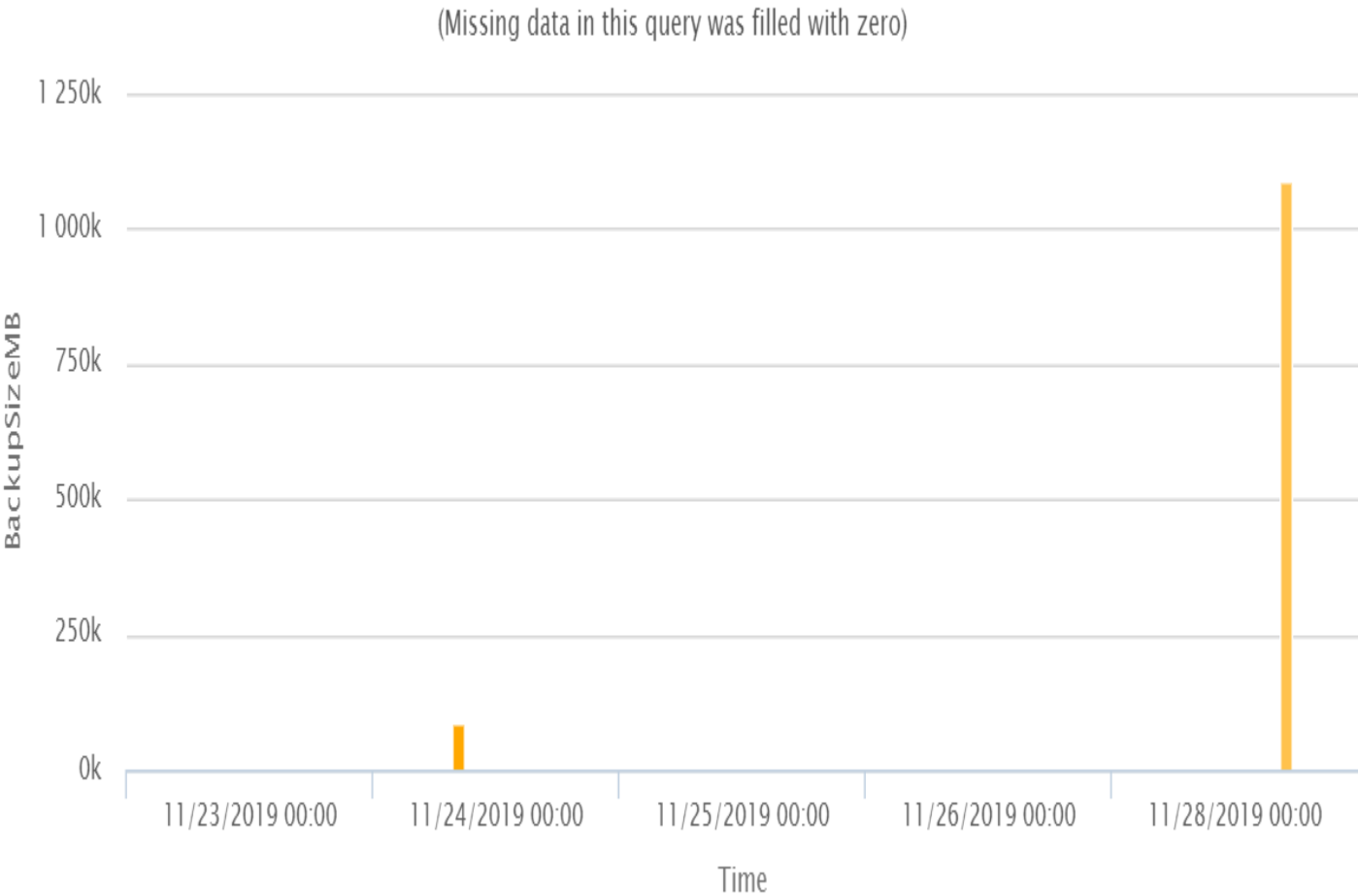
— CBI\_DATA LAKE (27ab22a1-224d-4a21-8e15-84c794122abf) — CBI\_STAGE (0072c38a-51d8-4775-a2b5-8f10b51b5d04)

— CMDB (51eb2811-b228-4f72-bd26-93d05a441b7)

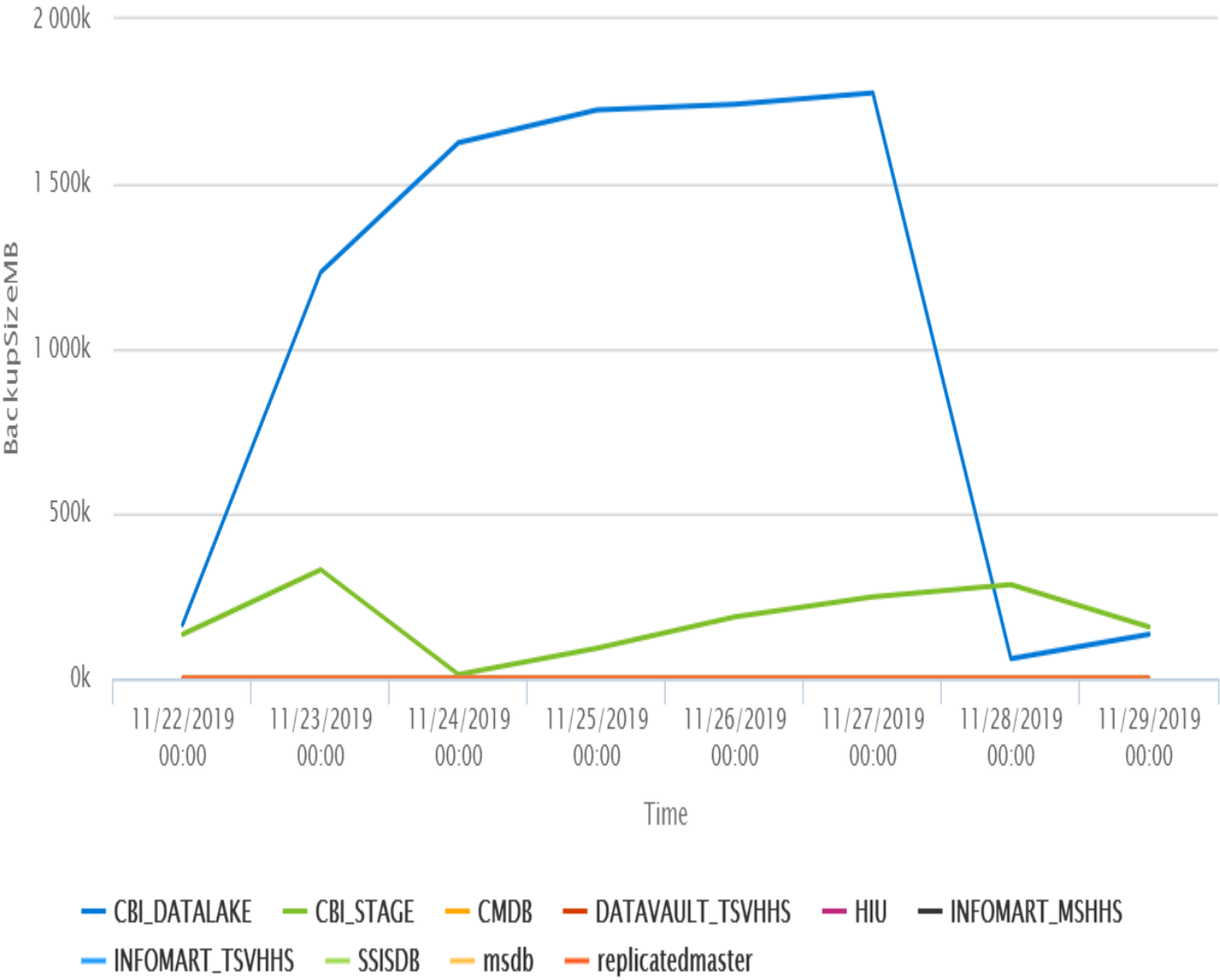
— DATAVAULT TSVHHS (e090b09e-6be0-4816-a495-3b027d9499c9) — HUI (95dd3dcd-27ec-4dc4-b3c1-fd3e6c2361b1)

▲ 1/3 ▼

PITR Backup Storage | Full backups per database (top 10)

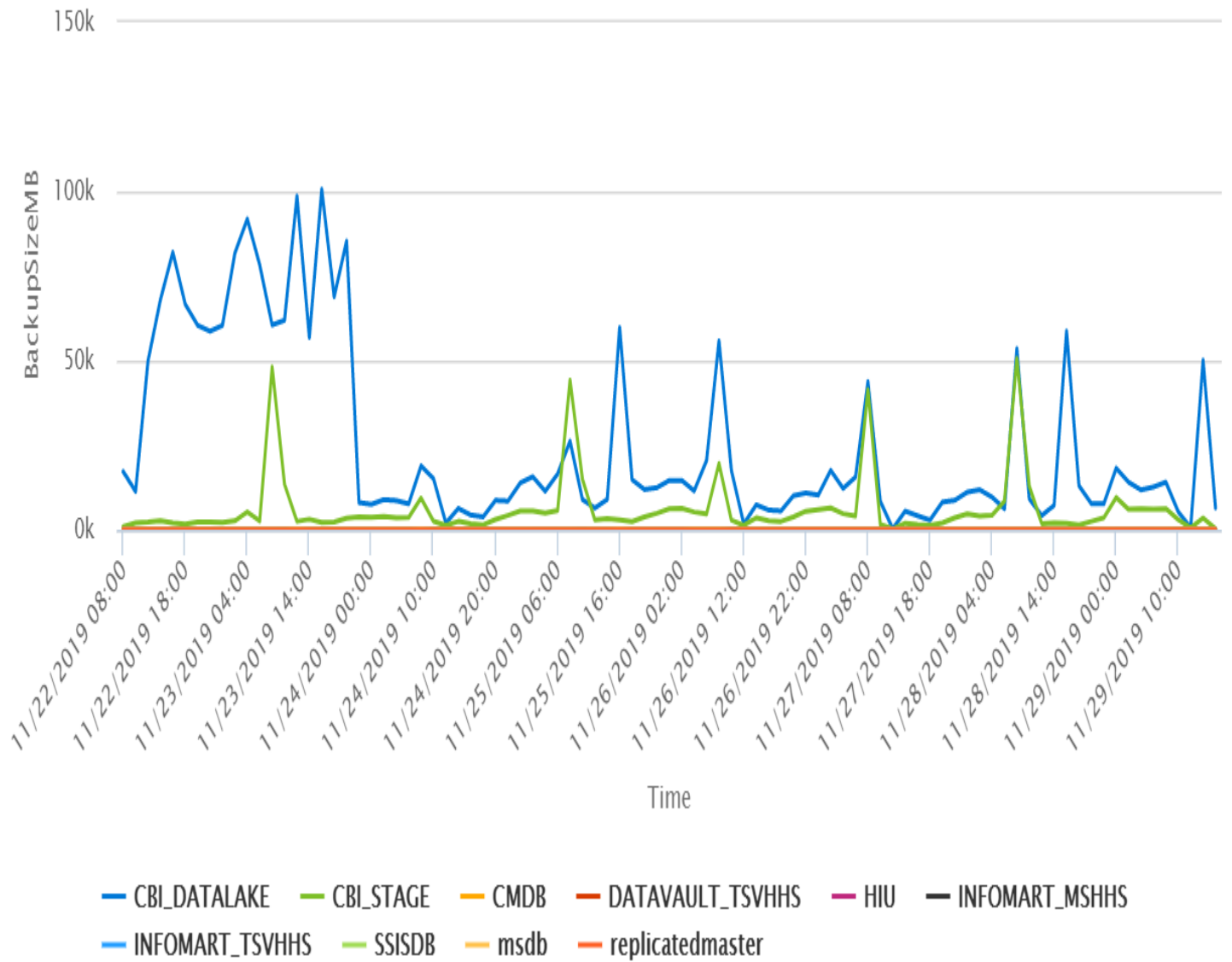


- INFOMART\_TSVHHS
- SSISDB
- CBI\_STAGE
- DATAVAULT\_TSVHHS
- HIU
- msdb
- replicatedmaster
- CMDB
- CBI\_DATALAKE
- INFOMART\_MSHHS



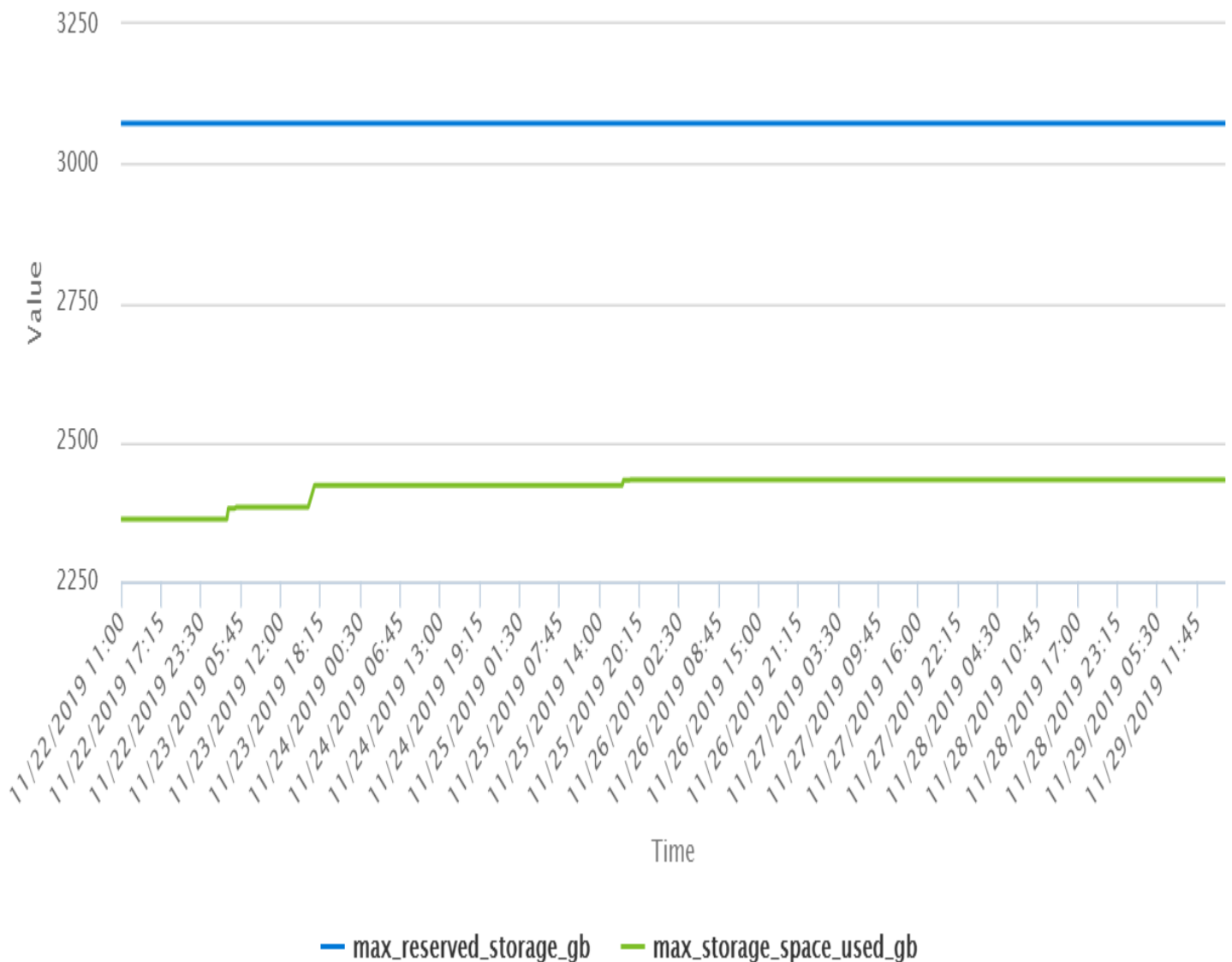
## PITR Backup Storage | Log backups per database (top 10)

Kusto Query ☺ ☹



## Storage reserved and used

Kusto Query ☺ ☹



## Check current database retention period using CMS

```
select managed_database_name, backup_retention_days
from managed_databases md
inner join managed_servers ms on md.managed_server_id = ms.managed_server_id
where ms.name = 'azuresqlmi'
```

## Known issue: Bug in the backup storage billing system continues

charging for dropped database on Managed Instance Nov 1st - Dec 31 affecting all customers worldwide.

MI PG has identified a bug in the backup billing charging logic which continues to charge the backup storage for dropped databases affecting all customers worldwide from Nov 1st until Dec 31st. If customers are deleting



their databases trying to save on the backup billing costs, there are no savings visible as the system continues billing for dropped databases. This results in very high backup storage bills for customers whereas no changes on the customer side are impacting the bill to go down. The **bug fix** has been **deployed worldwide as of Dec 31, 2019**. There will be **automated backup billing charge refunds for all customers affected**. The refunds are effectively expected to reflect the customers bills for **Feb 2020**.

To be very clear on this statement: only backup storage billing charges for deleted databases in the period Nov 1st - Dec 31st will be refunded, this does not mean the entire backup storage bill will be refunded as these are legitimate charges. This is to give heads up to all of our staff handling backup billing charges for MI customers. The public announcement will be sent to all customers affected mid-January 2020 through email to subscription owners, however you can share this information as of today with all customers complaining about high backup billing charges.

## Investigate backup chains and database operations

```
let MI = 'put_mi_name_here';
MonDwBilling
| where LogicalServerName =~ MI and AppTypeName contains "CL"
| where event == 'billing_metric_calculated'
| summarize argmax(TIMESTAMP, *) by storage_container_name, file_type, logical_database_id
| parse storage_container_name with something "/Full/" FullBackupTakenDate "T" FullBackupTakenTime "_0.bak"
| extend ChainDateTime = todatetime(strcat(FullBackupTakenDate, "T", replace("-", ":", FullBackupTakenTime)))
| project db_Id = toupper(logical_database_id), sizeGb = max(TIMESTAMP_size_in_bytes/ 1024/1024/1024,
file_type, ChainDateTime
| join kind=fullouter (
MonAnalyticsDBSnapshot
| where logical_server_name contains MI
| summarize max_timestamp = max(TIMESTAMP) by logical_database_id, logical_database_name, create_time
| extend db_Id = iff(logical_database_name == "msdb", "00000000-0000-0000-0000-000000000004",
logical_database_id)
| extend db_Id = iff(logical_database_name == "replicatedmaster", "00000000-0000-0000-0000-0000000032763",
db_Id)
| extend db_Id = iff(logical_database_name == "managed_model", "00000000-0000-0000-0000-0000000032760",
db_Id)
)
on db_Id
| extend Tombstoned = iff(max_timestamp < ago(2h) or isnull(max_timestamp), 1, 0)
| project db_Id, logical_database_id, logical_database_name, sizeGb, file_type, ChainDateTime, Tombstoned
| join kind=leftouter
(
MonManagementOperations
| where event == 'management_operation_success'
| where operation_result contains 'ManagedDatabaseId'
| where operation_type != 'DropManagedDatabase'
| extend d=parse_xml(operation_result)
| extend db_Id = toupper(tostring(d.OutputParameters.ManagedDatabaseId))
| extend TargetManagedDatabaseName = tostring(d.OutputParameters.TargetManagedDatabaseName)
| project db_Id, CreatedUsing = operation_type, TargetManagedDatabaseName, CreatedAt=
originalEventTimestamp
```

```

)
on db_Id
| join kind=leftouter
(
MonManagementOperations
| where event == 'management_operation_success'
| where operation_result contains 'ManagedDatabaseId'
| where operation_type == 'DropManagedDatabase'
| extend d=parse_xml(operation_result)
| extend db_Id = toupper(tostring(d.OutputParameters.ManagedDatabaseId))
| extend TargetManagedDatabaseName = tostring(d.OutputParameters.TargetManagedDatabaseName)
| project db_Id, DroppedUsing = operation_type, DroppedAt = originalEventTimestamp
)
on db_Id
| project-away db_Id1, db_Id2
// | where logical_database_name in ("", "")
| order by ChainDateTime asc

```

## RCA

In case you confirmed that deleted databases are driving the backup costs, share the following RCA with the customers:

The high backup billing costs are due to a bug in the backup billing logic for dropped databases affecting customers in the period Nov 1st – Dec 31st, 2019.

The issue is that there was a fault in the billing logic continuing to charge for dropped databases even after the retention period was out. For example, if you had a database with 7 day backup retention period and you deleted it, the system would wrongly continue billing for such backup for days 8, 9, and continuing indefinitely after the retention period has expired for a deleted database. This would result in the backup storage bill to pile up for customers, and especially in the case if you are continually adding and deleting databases.

The bug fix has been deployed worldwide and this is no longer an issue as of Jan 1st, 2020. For all affected customers there will be an official announcement sent via email to subscription owners by the end of January and automated refunds processed. You should expect to see this refund reflected on your bill the earliest in February or the latest in March 2020.

Please note that the refund would not necessarily be 100%, but only for the wrongly billed backup storage for deleted databases.

## Investigate Billed Quantity

```

//Accumulated storage consumption per database
MonDwBilling
| where LogicalServerName =~ '{ServerName}'
| where event == "billing_metric_calculated" and AppTypeName == "Worker.CL"
| extend size_mb = round((size_in_bytes/1024.0/1024.0),2)

```

```
| extend execution_time_sec = round((execution_time_msec/1000),2)
| project PreciseTimeStamp, AppName, logical_database_id=tolower(logical_database_name), file_type,
execution_time_sec, size_mb
| join kind=leftouter
(
MonAnalyticsDBSnapshot
| where logical_server_name =~ '{ServerName}'
| summarize arg_max(TIMESTAMP,*) by logical_database_id
| project logical_database_id=tolower(logical_database_id), backup_retention_days, logical_database_name
) on logical_database_id
| extend DatabaseName = iif(logical_database_name != '', logical_database_name, logical_database_id)
| distinct PreciseTimeStamp, DatabaseName, file_type, size_mb
| order by PreciseTimeStamp asc, DatabaseName asc
| summarize sum(size_mb) by DatabaseName, bin(PreciseTimeStamp,1h)
| where PreciseTimeStamp >= datetime(2019-11-12 01:00:00.0000000)
| render timechart
```

//SQL Database Managed Instance PITR Backup Storage billing meter for the instance

PostReportedUsage2

```
| where ServerName =~ '{ServerName}'
| where InfoFieldMeteredServiceType == 'SQL Database Managed Instance PITR Backup Storage'
| project-rename Date = EventDateTime
| project Date, ServerName, ServerId, UsageResourceQuantity
| order by Date asc
| render timechart
```

//Full backups per database

MonBackup

```
| where logical_server_name =~ '{ServerName}'
| where backup_type == 'Full'
| where event_type == 'BACKUP_METADATA_DETAILS'
| extend backup_size_MB = round(todecimal(backup_size)/1024/1024,2)
| extend uncompressed_backup_size_MB = round(todecimal(uncompressed_backup_size)/1024/1024,2)
| extend backup_size_GB = round(todecimal(backup_size)/1024/1024/1024,2)
| extend uncompressed_backup_size_GB = round(todecimal(uncompressed_backup_size)/1024/1024/1024,2)
| project PreciseTimeStamp, logical_database_id=tolower(logical_database_name), backup_size_MB,
backup_start_date, backup_end_date
| join kind=leftouter
(
MonAnalyticsDBSnapshot
| where logical_server_name =~ '{ServerName}'
| summarize arg_max(TIMESTAMP,*) by logical_database_id
| project logical_database_id=tolower(logical_database_id), logical_database_name
) on logical_database_id
| extend DatabaseName = iif(logical_database_name != '', logical_database_name, logical_database_id)
| project PreciseTimeStamp, DatabaseName, backup_size_MB, backup_start_date, backup_end_date
| extend backup_duration = todatetime(backup_end_date) - todatetime(backup_start_date)
| order by PreciseTimeStamp asc
```

```
// | project PreciseTimeStamp, DatabaseName, backup_size_MB
```

```
// | render timechart
```

```
//Differential backups per database
```

```
MonBackup
```

```
| where logical_server_name =~ '{ServerName}'
```

```
| where backup_type == 'Diff'
```

```
| where event_type == 'BACKUP_METADATA_DETAILS'
```

```
| extend backup_size_MB = round(todecimal(backup_size)/1024/1024,2)
```

```
| extend uncompressed_backup_size_MB = round(todecimal(uncompressed_backup_size)/1024/1024,2)
```

```
| extend backup_size_GB = round(todecimal(backup_size)/1024/1024/1024,2)
```

```
| extend uncompressed_backup_size_GB = round(todecimal(uncompressed_backup_size)/1024/1024/1024,2)
```

```
| project PreciseTimeStamp, logical_database_id=tolower(logical_database_name), backup_size_MB,  
backup_start_date, backup_end_date
```

```
| join kind=leftouter
```

```
(
```

```
MonAnalyticsDBSnapshot
```

```
| where logical_server_name =~ '{ServerName}'
```

```
| summarize arg_max(TIMESTAMP,*) by logical_database_id
```

```
| project logical_database_id=tolower(logical_database_id), logical_database_name
```

```
) on logical_database_id
```

```
| extend DatabaseName = iif(logical_database_name != '', logical_database_name, logical_database_id)
```

```
| project PreciseTimeStamp, DatabaseName, backup_size_MB, backup_start_date, backup_end_date
```

```
| extend backup_duration = todatetime(backup_end_date) - todatetime(backup_start_date)
```

```
| order by PreciseTimeStamp asc
```

```
| project PreciseTimeStamp, DatabaseName, backup_size_MB
```

```
| render timechart
```

```
//Log backups per database
```

```
MonBackup
```

```
| where logical_server_name =~ '{ServerName}'
```

```
| where backup_type == 'Log'
```

```
| where event_type == 'BACKUP_METADATA_DETAILS'
```

```
| extend backup_size_MB = round(todecimal(backup_size)/1024/1024,2)
```

```
| extend uncompressed_backup_size_MB = round(todecimal(uncompressed_backup_size)/1024/1024,2)
```

```
| extend backup_size_GB = round(todecimal(backup_size)/1024/1024/1024,2)
```

```
| extend uncompressed_backup_size_GB = round(todecimal(uncompressed_backup_size)/1024/1024/1024,2)
```

```
| project PreciseTimeStamp, logical_database_id=tolower(logical_database_name), backup_size_MB,  
backup_start_date, backup_end_date
```

```
| join kind=leftouter
```

```
(
```

```
MonAnalyticsDBSnapshot
```

```
| where logical_server_name =~ '{ServerName}'
```

```
| summarize arg_max(TIMESTAMP,*) by logical_database_id
```

```
| project logical_database_id=tolower(logical_database_id), logical_database_name
```

```
) on logical_database_id
```

```
| extend DatabaseName = iif(logical_database_name != '', logical_database_name, logical_database_id)
```

```
| project PreciseTimeStamp, DatabaseName, backup_size_MB, backup_start_date, backup_end_date
```

```
| extend backup_duration = todatetime(backup_end_date) - todatetime(backup_start_date)
```

```
| order by PreciseTimeStamp asc
| project PreciseTimeStamp, DatabaseName, backup_size_MB
| render timechart

//Storage
MonManagedInstanceResourceStats
| where server_name =~ '{ServerName}'
| project PreciseTimeStamp, reserved_storage_mb, storage_space_used_mb
| extend reserved_storage_gb = reserved_storage_mb/1024.0
| extend storage_space_used_gb = storage_space_used_mb/1024.0
| summarize max(reserved_storage_gb), max(storage_space_used_gb) by bin(PreciseTimeStamp,15m)
| render timechart
```

**How good have you found this content?**

