

Flexible Server Performance

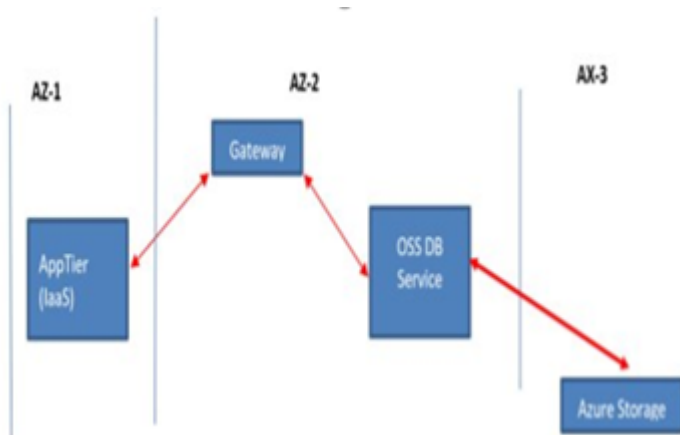
Last updated by | Hamza Aqel | Oct 5, 2022 at 5:59 AM PDT

Colocation:

In Azure Database for PostgreSQL Single Server, the challenge we have is when we provision Azure DB for PostgreSQL in a region, the Region has many availability Zones. Availability Zone (AZ) is like a Data Center. Underneath an AZ we can have multiple Datacenters. It is a logical entity that represents those Data centers. In a region you can have 1 or 3 Availability Zones.

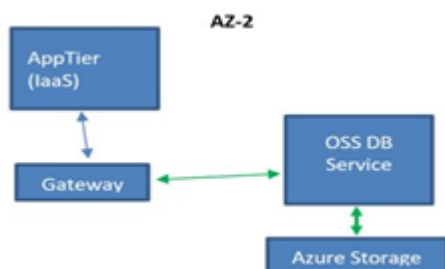
Eg: West Europe has 3 AZ's. When you provision a server (Postgres Single Server or Sterling), we do not have the control where the Database, application Tier and Azure Storage would land.

Worst Case:



In the worst-case Scenario: App Tier, Database Service and Azure Storage can land in Different Availability Zones. On an average you can assume interzone latency is the order of 1-2 ms. If we do a connection from the App, it should cross the AZ boundary and there is some latency. Similarly there is some latency between DB Compute and Storage if they land in different AZ's.

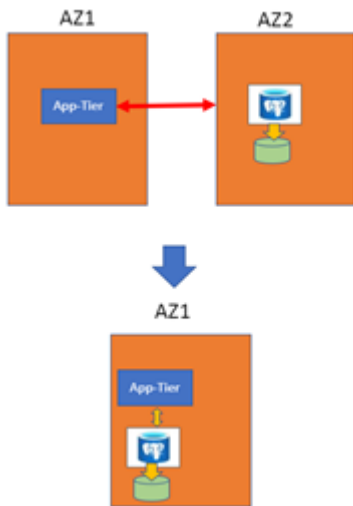
Best Case:



The best case is if App Tier, Database Services and Azure Storage are sitting in the same Availability Zone. Azure's guidelines say that if you're in same AZ, the average latency is less than 1ms.

Based on numerous feedbacks from customers, this limitation is addressed in the new Flexible Server deployment.

We now have the Ability to collocate app-tier in the same AZ as Flexible Server.



Why Colocation?

Achieve < 1ms latency between app-tier and DB

How can we achieve colocation?

Specify App-tier AZ when provisioning DB

Also, In Flexible Server deployment, we do not have the Gateway, so we avoid an extra Hop. The overall expected latency with flexible server is under 1 ms.

Microsoft Azure (Preview)

Report a bug

Search resources, services, and docs (G+ /)


[Home](#) > [Azure Database for PostgreSQL servers](#) > [Select Azure Database for PostgreSQL deployment option](#) >


Flexible server (Preview)


Microsoft - preview


Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * 

Abhisheksub 


Resource group * 


Abhishekresourcegroup 


[Create new](#)


Server details


Enter required settings for this server, including picking a location and configuring the compute and storage resources.


Server name * 

myflexserver1 

Region * 

East US 

Availability zone 

No preference 


No preference


1


2

3


[Configure server](#)

High Availability 


PostgreSQL version * 

Compute + storage 


Administrator account

Admin username * 

Enter server admin login name

Password * 

Enter password

Confirm password * 

Confirm the above password

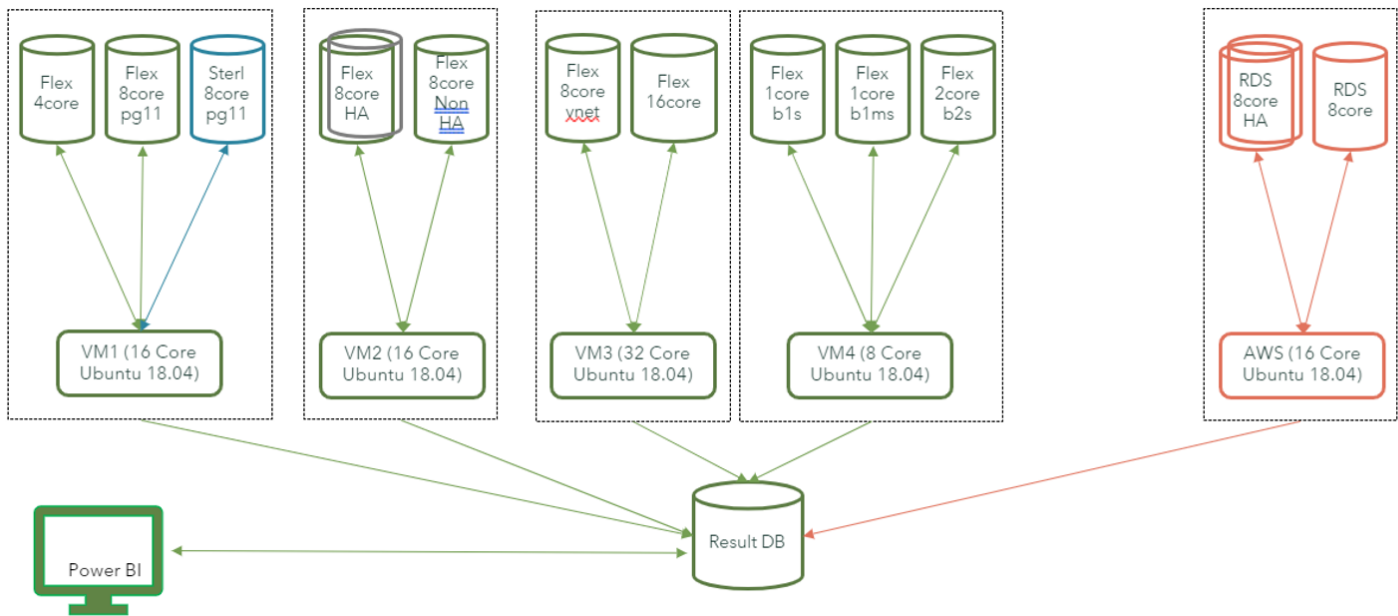
Review + create

Next : Networking >

Performance:

After Reviewing a number of feedbacks from customers who have made comparisons between Single Server and Postgres on VM's/ Postgres on Competitors Cloud platforms, a number of limitations have been addressed in the new Flexible Server deployment.

Below are the Performance Benchmarking tests (Pgbench) performed on different SKU's on Azure Flexible Server and also with Similar Vcore/ SKU on our Competitor's cloud (RDS).



- Current Test Suite comprises of following components
- Client VMs (Ubuntu VMs in Azure and AWS)
- Target Server (PG Servers in Azure and AWS)
- Result DB to store test results
- All Azure Clients and Servers are deployed in East US2 EUAP - AZ 3
- All AWS Clients and Servers are deployed in us-east-2c AZ
- All clients runs on Ubuntu 18.04.1 and have cores twice the server core size to avoid any client-side throttling • All Azure clients have accelerated networking enabled
- PG Bench 10.12 is used for all test cases
- Following PG Bench settings have been updated like below:
 - Checkpoint_Completion_Target increased from 0.5 to 0.8
 - Max_Wal_Size decreased from 32768 to 8192 (in MB)
 - Wal_compression is enabled

Client Details:

Client Name	vCore
flexserverpg-canary-clientvm1-16core	16
flexserverpg-canary-clientvm2-16core	16
flexserverpg-canary-clientvm3-32core	32
flexserverpg-canary-clientvm4-8core	8
aws_client_16core_useast_2c	16

Flexible Server and AWS RDS SKU's/ Configuration:

Server Name	Server Edition	vCore	Storage
flexserverpg-canary-perf-b1s-1core	Burstable	1	512GB
flexserverpg-canary-perf-b1ms-1core	Burstable	1	512GB
flexserverpg-canary-perf-b2s-2core	Burstable	2	512GB
flexserverpg-canary-perf-gp-4core	GP	4	2TB
flexserverpg-canary-perf-gp-8core	GP	8	2TB
flexserverpg-canary-perf-gp-8core-ha	GP with HA	8	2TB
flexserverpg-canary-perf-gp-8core-vnet	GP with VNET	8	2TB
flexserverpg-canary-perf-gp-8core-pg11	GP with PG11	8	2TB
flexserverpg-canary-perf-gp-16core	GP	16	2TB
sterlingpg-canary-perf-gp-8core-pg11	GP (Sterling PG 11)	8	2TB
rds-pg12-3-standard-8core-ha	GP (RDS with HA)	8	2TB

Following tests are executed for each SKU.

- For RO/RW workload, first 15 min of data is ruled out as Warm up. Data from 15th to 20th minute is used for measurements.
- Explicit checkpoint is issued before the measurement run to avoid discrepancies between any given test.

- Overall test run for all Servers and cases together is around 18 hours. With 4 clients involved, our turnaround time is about 5-6 hours.

- Below data provides total time it takes to initialize and test any given SKU/Test combination.

Test Details:

S.No	vCore	Test Type	No. of Clients / Threads	Scale	Warmup Duration	Measurement Duration
1	n	Latency-Select1	1	0	0	300
2	n	Latency-Select1NPPS	$(8 * n)$	0	0	300
3	n	<u>RO_FullyCached</u>	$(8 * n)$	$(90 * n) / 2$	900	300
4	n	<u>RO_BorderLine</u>	$(8 * n)$	$(360 * n) / 2$	900	300
5	n	<u>RW_FullyCached</u>	$(8 * n)$	$(90 * n) / 2$	900	300

****Fully Cached means data is entirely in Memory.**

Border Line means mostly in Memory but some Reads do go/hit the disk.**

S.no	Test Type	4core	16core	8core	8core-vnet	flex-8core-pg11	Sterling-8core-pg11	b1s	b1ms	b2s	rds-8core-ha	flex-8core-ha
1	select1	5	5	5	5	5	5	5	5	5	5	5
2	select1npps	5	5	5	5	5	5	5	5	5	5	5
3	<u>RO_FullyCache(Warmup)</u>	17	21	18	18	18	22	18	18	20	16	17
4	<u>RO_FullyCache(Measurement)</u>	5	5	5	5	5	5	5	5	5	5	5
5	<u>RO_Borderline(Warmup)</u>	28	41	36	33	33	40	28	28	38	20	33
6	<u>RO_Borderline(Measurement)</u>	5	5	5	5	5	5	5	5	5	5	5
7	<u>RW_FullyCache(Warmup)</u>	17	21	17	17	17	22	18	18	19	16	17
8	<u>RW_FullyCache(Measurement)</u>	5	5	5	5	5	5	5	5	5	5	5
9	<u>RW_FullyCache_16xcore(Warmup)</u>										16	17
10	<u>RW_FullyCache_16xcore(Measurement)</u>										5	5
	Total Time	87	108	96	93	93	109	89	89	102	98	114

Performance Comparison:

Test Name	FlexServer PG11 (8vC)	Sterling PG11(8vC)
Select1	5400	1290
Select1NPPS	81500	71500
RO_FullyCached	64000	56000
RO_BorderLine	58000	33500
RW_FullyCached	7310	4200

Test Name	FlexServer 8Core HA	RDS 8Core HA
Select1	5010	8500
Select1NPPS	83000	164000
RO_FullyCached	64344	119000
RO_BorderLine	58213	108000
RW_FullyCached	4120	10200

Test Name	FlexServer 8Core	RDS 8Core
Select1	5400	8600
Select1NPPS	79000	165000
RO_FullyCached	62000	120000
RO_BorderLine	58000	110000
RW_FullyCached	7310	12800

Test Name	FlexServer 8core VNET	FlexServer 8Core
Select1	3800	5400
Select1NPPS	86100	79000
RO_FullyCached	67200	62000
RO_BorderLine	60523	58000
RW_FullyCached	7500	7310

Test Name	FlexServer PG11 8core	FlexServer PG12 8core
Select1	5400	5400
Select1NPPS	81500	79000
RO_FullyCached	64000	62000
RO_BorderLine	58000	58000
RW_FullyCached	7310	7310

Test Name	FlexServer B2S	FlexServer GP 2Core
Select1	5800	2200
Select1NPPS	33800	20000
RO_FullyCached	23500	14000
RO_BorderLine	1300	11900
RW_FullyCached	1200	1600

Test Name	FlexServer 8core MO	FlexServer 8Core GP
Select1	5400	5400
Select1NPPS	79000	79000
RO_FullyCached	63000	62000
RO_BorderLine	61000	58000
RW_FullyCached	7000	7310

Test Name	FlexServer 8Core No-HA	FlexServer 8Core HA
Select1	5400	5010
Select1NPPS	79000	83000
RO_FullyCached	62000	64344
RO_BorderLine	58000	58213
RW_FullyCached	7310	4120

Test Name	FlexServer	Sterling	RDS
Select1 (Latency)	0.20ms	0.80ms	0.11ms

The above picture shows different comparisons between Single Server/ Flexible Server, Flexible Server/ AWS RDS and also with High Availability Enabled.

In the first table for example, we are comparing an 8 Vcore Single Server vs 8 Vcore Flexible Server.

You can notice, the select 1 performance improved drastically (4-5x) times. Other workload tests also see a huge improvement.

In the first row third column, you can see comparison with AWS RDS.

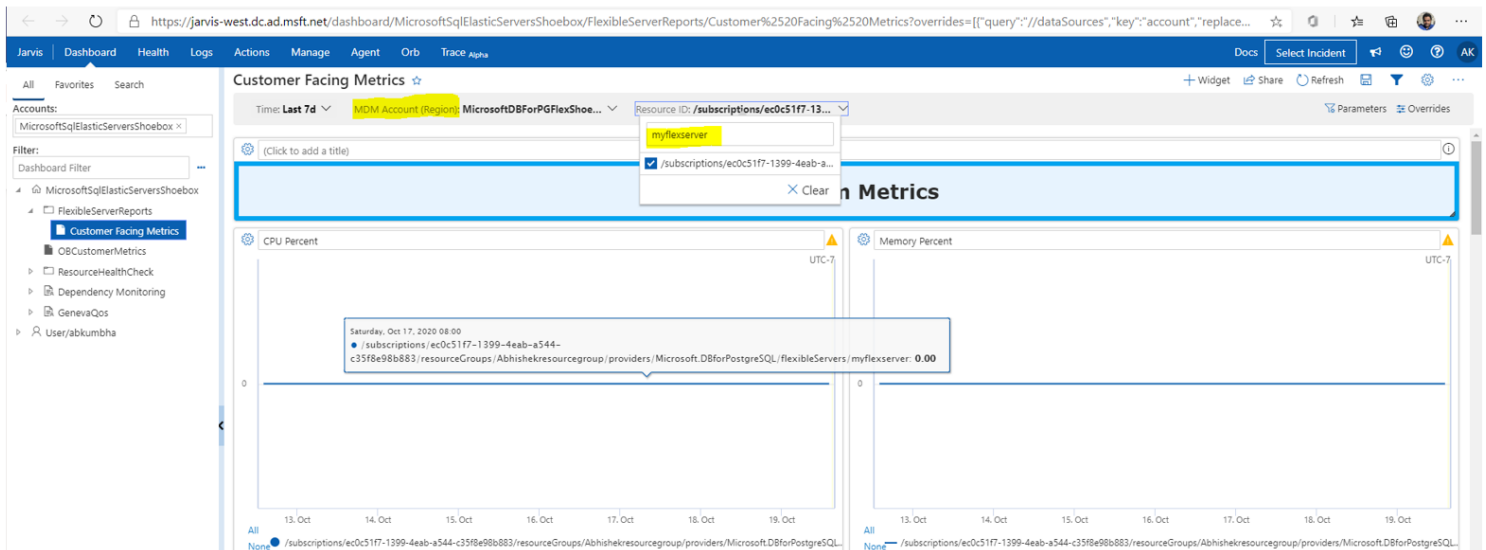
Other tables have information between Versions/ HA/ with-without Vnet etc. Please Note that this information is internal, and we need not share all these numbers with customers.

Performance Metrics can be measured using

1. Jarvis
2. ASC
3. Kusto
4. XTS

Jarvis:

<https://aka.ms/FlexServerMetricsDash> 



MDM Account is the place where metrics are actually stored. The MDM is system from where Portal exposes Metrics to Customers.

Select the Region under MDM Account

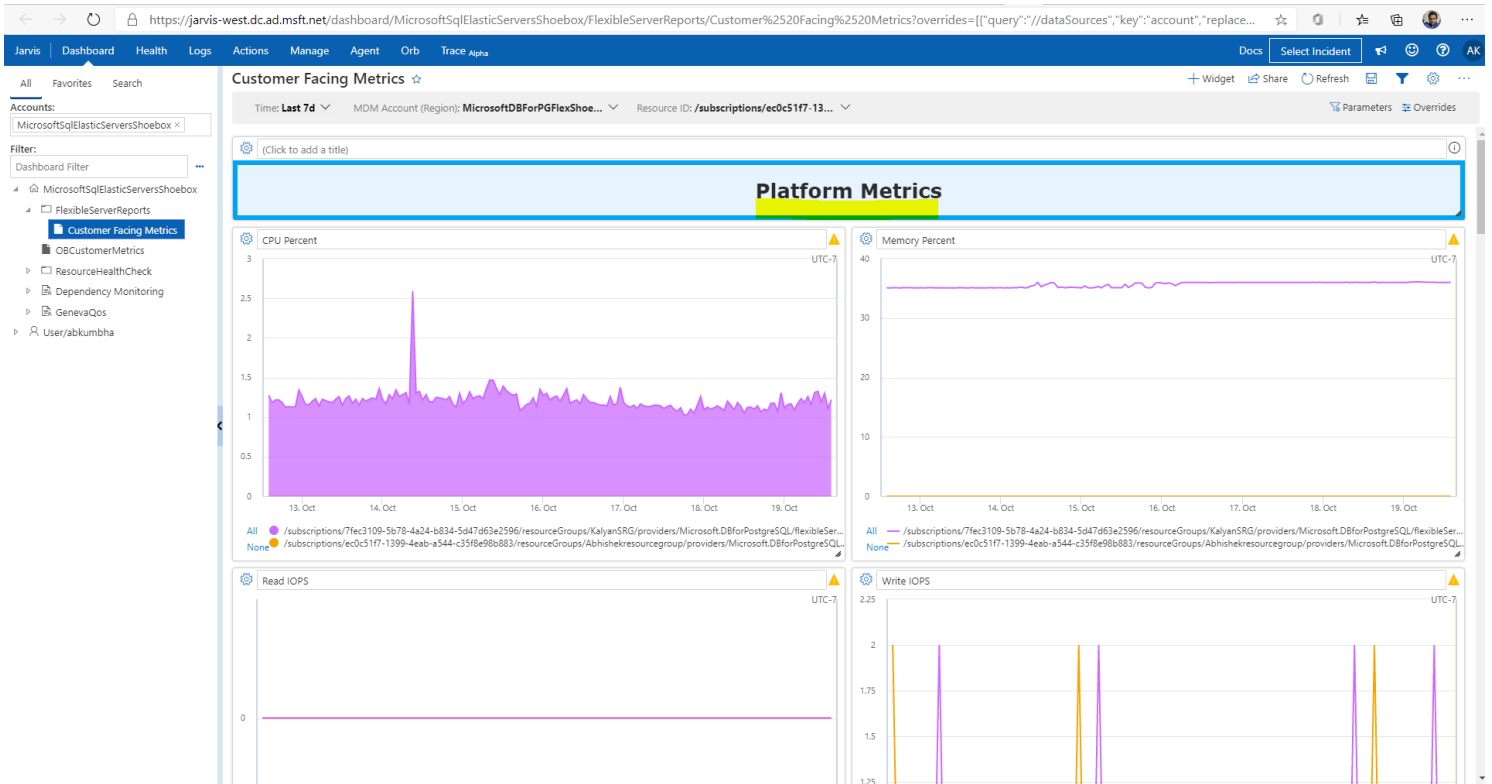
Under the Resource ID, Start typing your resource name and it should pull up the Resource URI.

In this Monitor, You have several Metrics for measuring Performance:

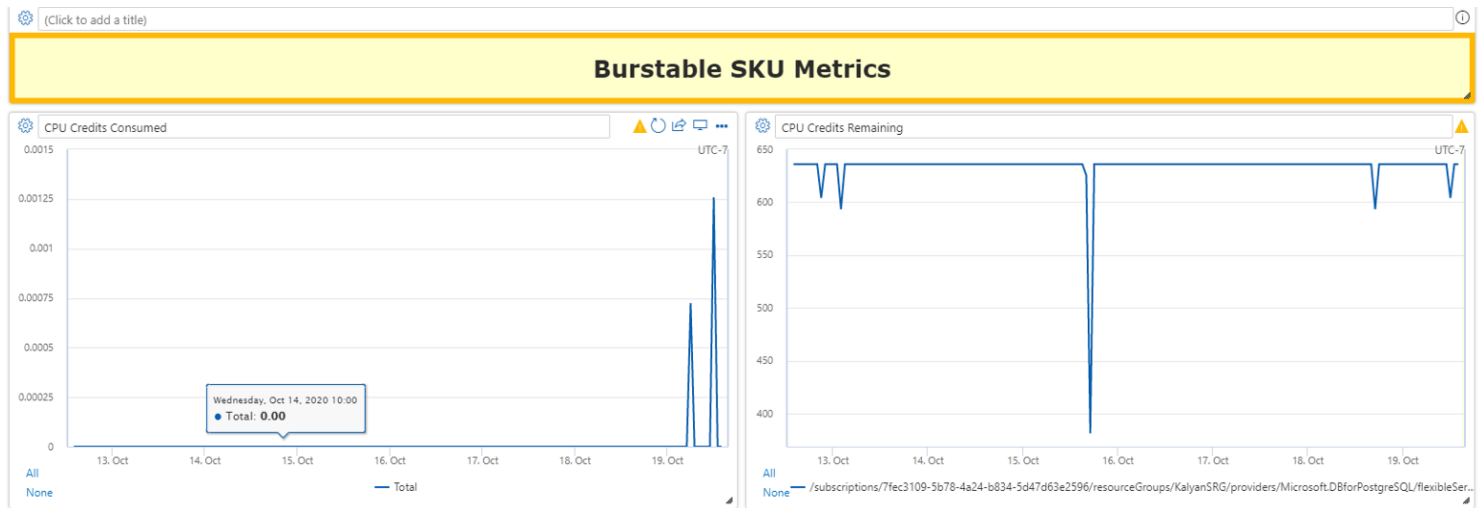
Postgres Flexible Server Metrics Dashboard: <https://aka.ms/FlexServerMetricsDash>

Platform Metrics: Platform Metrics is Metrics of the VM in which PG is running.

1. CPU Percent
2. Memory Percent
3. Read IOPS
4. Write IOPS
5. Read Throughput
6. Write Throughput
7. IOPS
8. Disk Queue Depth



There are Metrics like CPU Credits Consumed and Remaining for Burstable Servers.

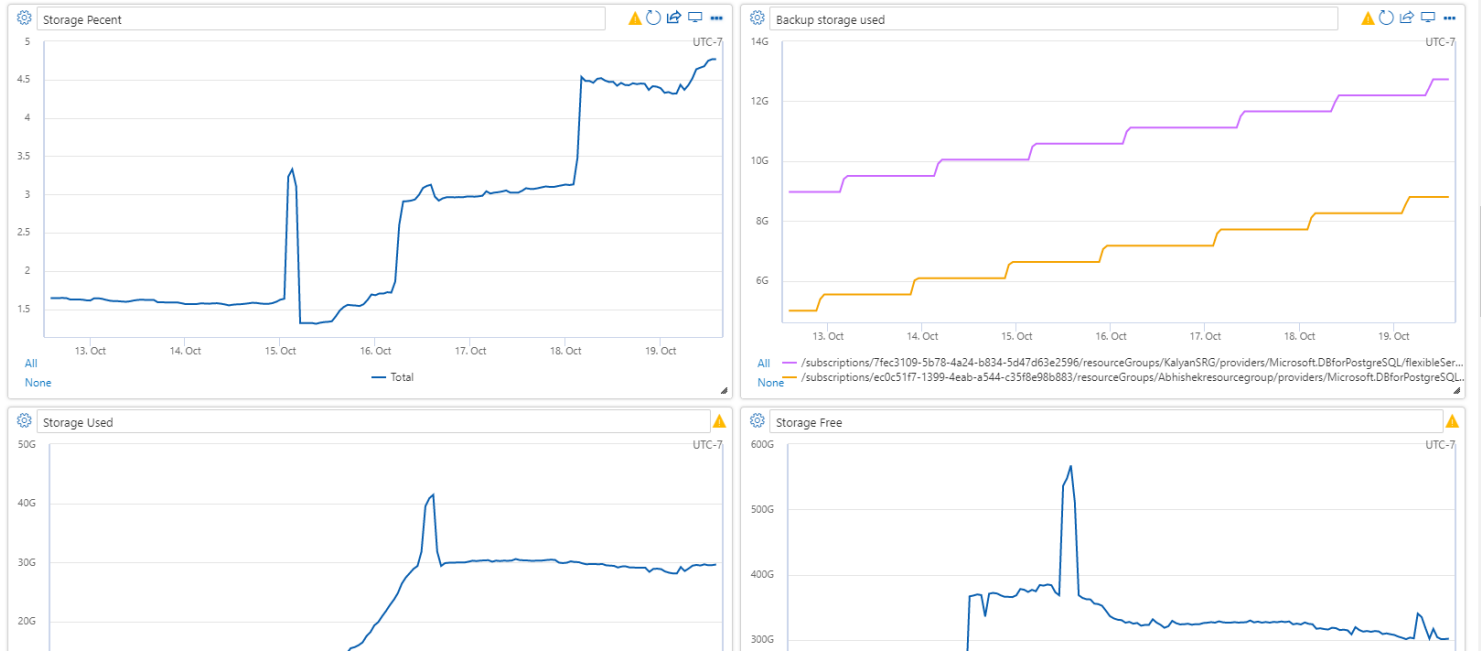


You also have Storage, Connectivity and Networking Metrics for the Selected Server.

Storage Metrics:

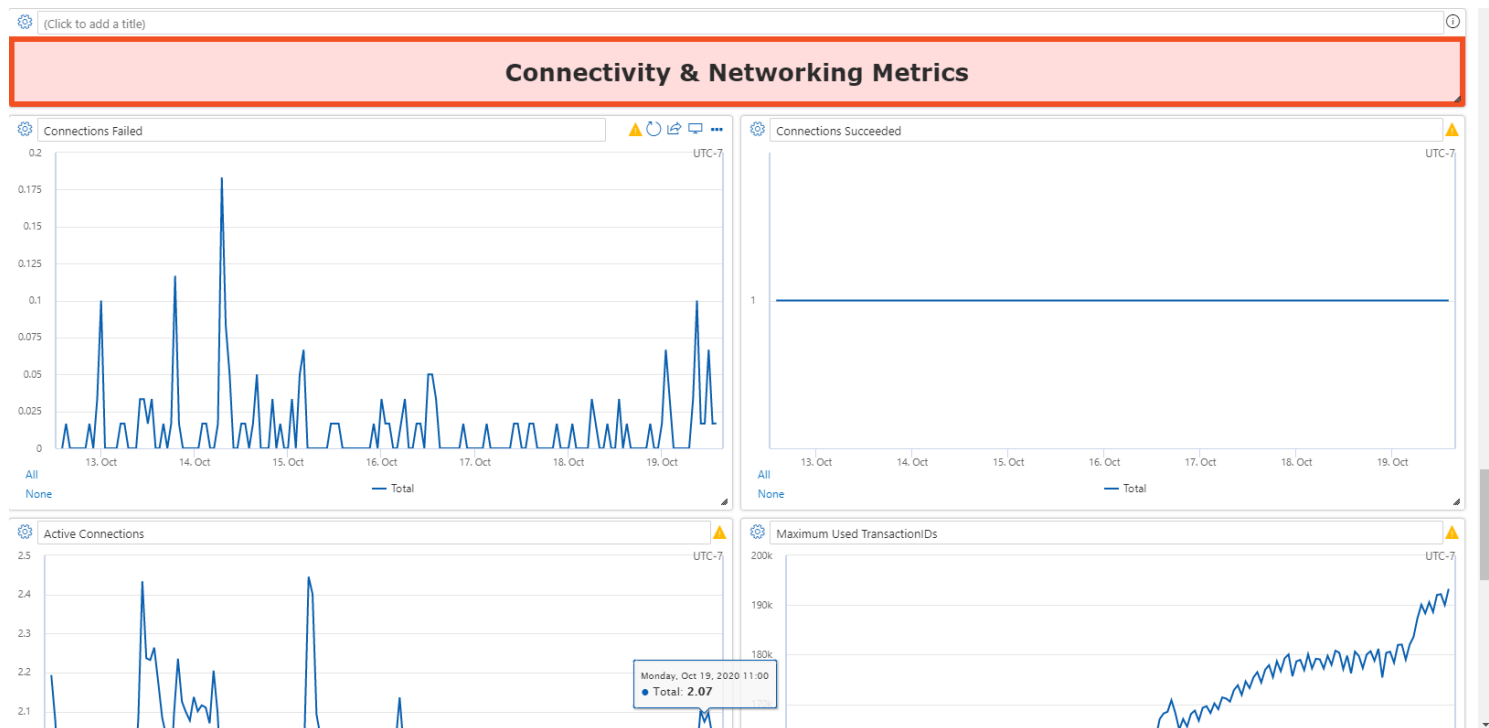
1. Storage Percent.
2. Backup storage Used
3. Storage Used
4. Storage Free
5. Transaction Log Storage used

Storage Metrics




Connectivity and Networking Metrics:

1. Connections Failed
2. Connections Succeeded
3. Active Connections
4. Maximum Used TransactionID's
5. Network Bytes Egress
6. Network Bytes Ingress

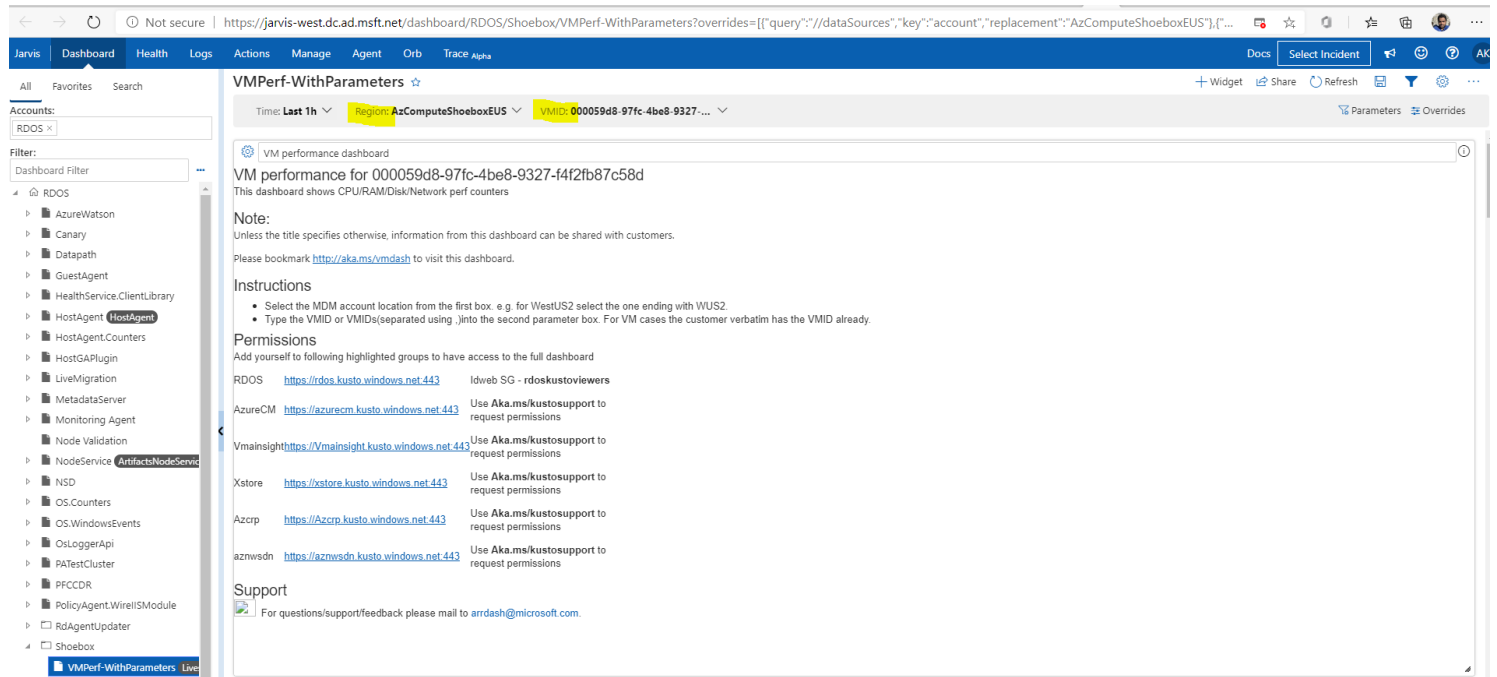


These are all the Metrics customers could see on their Portal.

ASC will expose this link going forward where MDM account and Resource ID will be Preselected for you.

Azure VM Metrics Dashboard: <https://aka.ms/VmDash> 

This will show Metrics of the VM on which PG is Running.



The screenshot shows the Jarvis tool interface with the 'VMPerf-WithParameters' dashboard. The dashboard displays VM performance metrics for a specific VM (000059d8-97fc-4be8-9327-f4f2fb87c58d) in the AzComputeShoeboxEUS region. The dashboard includes sections for Note, Instructions, Permissions, and Support.

Note: Unless the title specifies otherwise, information from this dashboard can be shared with customers. Please bookmark <http://aka.ms/vmdash> to visit this dashboard.

Instructions:

- Select the MDM account location from the first box. e.g. for WestUS2 select the one ending with WUS2.
- Type the VMID or VMIDa(separated using _ into the second parameter box. For VM cases the customer verbatim has the VMID already.

Permissions:

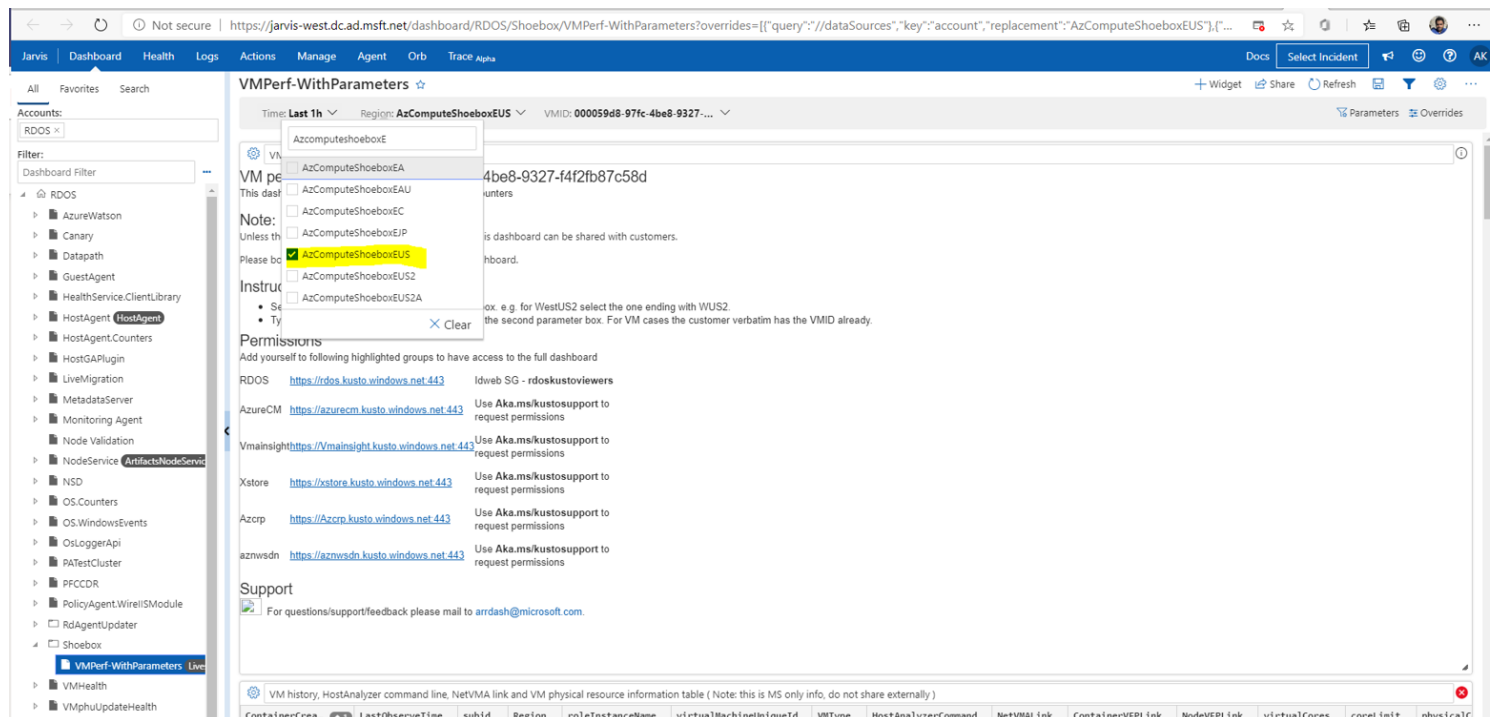
Add yourself to following highlighted groups to have access to the full dashboard

- RDOS <https://rdos.kusto.windows.net.443> Idweb SG - rdoskustoviewers
- AzureCM <https://azurecm.kusto.windows.net.443> Use Aka.ms/kustosupport to request permissions
- Vmainsight <https://vmainsight.kusto.windows.net.443> Use Aka.ms/kustosupport to request permissions
- Xstore <https://xstore.kusto.windows.net.443> Use Aka.ms/kustosupport to request permissions
- Azcorp <https://azcorp.kusto.windows.net.443> Use Aka.ms/kustosupport to request permissions
- aznwsdn <https://aznwsdn.kusto.windows.net.443> Use Aka.ms/kustosupport to request permissions

Support: For questions/support/feedback please mail to arrdash@microsoft.com.

AccountName is AzComputeShoebox followed by Region Name.

Example: If region is EastUS:



The screenshot shows the Jarvis tool interface with the 'VMPerf-WithParameters' dashboard. A dropdown menu is open, showing a list of MDM account locations. The selected account is 'AzComputeShoeboxEUS'.

VM performance dashboard

VM performance for 000059d8-97fc-4be8-9327-f4f2fb87c58d

This dashboard shows CPU/RAM/Disk/Network perf counters

Note: Unless the title specifies otherwise, information from this dashboard can be shared with customers. Please bookmark <http://aka.ms/vmdash> to visit this dashboard.

Instructions:

- Select the MDM account location from the first box. e.g. for WestUS2 select the one ending with WUS2.
- Type the VMID or VMIDa(separated using _ into the second parameter box. For VM cases the customer verbatim has the VMID already.

Permissions:

Add yourself to following highlighted groups to have access to the full dashboard

- RDOS <https://rdos.kusto.windows.net.443> Idweb SG - rdoskustoviewers
- AzureCM <https://azurecm.kusto.windows.net.443> Use Aka.ms/kustosupport to request permissions
- Vmainsight <https://vmainsight.kusto.windows.net.443> Use Aka.ms/kustosupport to request permissions
- Xstore <https://xstore.kusto.windows.net.443> Use Aka.ms/kustosupport to request permissions
- Azcorp <https://azcorp.kusto.windows.net.443> Use Aka.ms/kustosupport to request permissions
- aznwsdn <https://aznwsdn.kusto.windows.net.443> Use Aka.ms/kustosupport to request permissions

Support: For questions/support/feedback please mail to arrdash@microsoft.com.

VM history, HostAnalyzer command line, NetVMA link and VM physical resource information table (Note: this is MS info only, do not share externally)

ContainerCrea	LastObserveTime	subid	Region	roleInstanceName	virtualMachineUniqueid	VMType	HostAnalyzerCommand	NetVMAlink	ContainerVFPLink	NodeVFPLink	virtualCores	coreLimit	physicalIC

VM ID: To get the VM ID, please use the below link

Execute: [Web] [Desktop] [Web (Lens)] [Desktop (SAW)] <https://azurecm.kusto.windows.net/AzureCM> 

LogContainerSnapshot | where PreciseTimeStamp >= ago(1d) and roleInstanceName contains "VMNAME_FROM_ASC" | distinct virtualMachineUniqueld

You can get VM Name from ASC under Server info:

Server Info

High level information about this server

Drag a column header and drop it here to group by that column	
PropertyName	PropertyValue
Server Name	myflexserver
Server Type	flexibleServers
Server Edition	GeneralPurpose
Server State	Succeeded
Server SKU	Standard_D2s_v3
Virtual Cores	2
Storage Limit (MiB)	65536
Storage Limit (GiB)	64
Customer Subscription Id	ec0c51f7-1399-...
Customer Resource Group Name	Abhishekresourcegroup
Server Location	EASTUS
Cluster Short Name	ProdEus1a
Virtual Machine Name	b4e75876d76e...
Resource Uri	/subscriptions/.../resourceGroups/Abhishekresourcegroup/providers/Microsoft.DBforPostgreSQL/flexibleServers/myflexserver
Create Time	10/2/2020 6:01:28 AM
Microsoft Subscription Id	A974D7C8-...
Microsoft Management Service Cluster	tr1736.eastus1-a.worker.database.windows.net

VM ID:

Azure Data Explorer

Query

Filter...

1 LogContainerSnapshot | where PreciseTimeStamp >= ago(1d) and roleInstanceName contains "b4e75876d76e"

2 | distinct virtualMachineUniqueld

Table 1

virtualMachineUniqueld
ae09d952-9da7-4b45-a4ce-106cebaa6e05

Kusto Tables to Query for Flexible Server:

Flexible Server	Single Server
MonOrcasBreadthCMSSnapshot	MonAnalyticsElasticServersSnapshot
MonOrcasBreadthResourceProvider	MonManagementResourceProvider
AlrOrcasBreadthRp	AlrManagement
MonOrcasBreadthRp, MonOrcasBreadthRpExceptions,PiiOrcasBreadthRpExceptions	MonManagement, MonManagementE
MonOrcasBreadthDirectorEvents	/
MonOrcasBreadthActorEvents	/
/	Monlogin
WinFabLogs	WinFabLogs
OBvmagentlog, piiOBvmagentsidecarlog, OBvmagentsidecarpgsql, piiOBvmagentsidecarpgsql	MonRdmsInstanceAgent
MonPgLogs, PiiPgLogs	MonRdmsPgSqlLaunchSetup, MonRdmsPgSqlSandbox, MonRdmsPg PiiRdmsPgServer
PiiPgLogsShoebox	PiiRdmsPgServerShoebox
MonOBDockerContainerEvents	/
MonOBDockerStats	/
MonOBVmStats	MonRdmsServerMetrics, MonRgLoad
MonOBStorageStats	MonRdmsServerMetrics, MonRgLoad, MonDmIoVirtualFileStats (SBS), MonDmOsDatabasesCounters (SBS)

Flexible Server	Single Server
MonOBPgWalReplica	/
MonOBPgSqlSessionsStatus	MonOBPgSqlSessionsStatus
MonOBPgSqlDBSizes	MonDmPgSqlDBSizes
MonOBPgSqlBuffers	MonDmPgSqlBuffers
MonOBPgSqlCacheHit	MonDmPgSqlCacheHit
MonOBPgSqlTransactionStats	MonDmPgSqlTransactionStats
MonOBPgSqlAppConnection	MonDmPgSqlAppConnection
MonOBPgSqlConnectionStats	MonRdmsServerMetrics,MonDmPgSql
MonOBPgSqlTablesIOStats	/
MonOBPgSqlTotalUserQueries	MonOBPgSqlTotalUserQueries
MonOBPgSqlStatStatements	MonDmPgSqlStatStatements
MonOBPgSqlSessions	MonDmPgSqlSessions
MonOBPgSqlConnections	MonDmPgSqlConnections
MonOBPgSqlFunctions	MonDmPgSqlFunctions
MonOBPgSqlIndexesStats	/
MonOBPgSqlIndexesEfficiency	MonDmPgSqlIndexesEfficiency

Flexible Server	Single Server
MonOBPgSqlTablesStats	/
MonOBPgSqlExtensionStats	MonDmPgSqlExtensionStats
MonOBPgSqlSequentialScans	MonDmPgSqlSequentialScans
MonOBPgSqlXlogFileCount	/
MonOBPgSqlReplicationStats	MonDmPgSqlReplicationStatsPrimary

Connection Errors:

Since Flexible Server doesn't have a Gateway, there is no MonLogin.

Most Errors Should be correlating with psql Errors that thrown from client. Please refer below link.

PostgreSQL Error Codes: <https://www.postgresql.org/docs/11/errcodes-appendix.html> 

Some useful Kusto:

MonOBPgSqlConnectionStats | where LogicalServerName contains "myflexserver" | where TIMESTAMP > ago (1h)

Container Metrics: MonOBDockerStats | where LogicalServerName contains "myflexserver" //| where TIMESTAMP > datetime(2020-05-04 08:00:00.0000000) and TIMESTAMP < datetime(2020-05-04 14:59:00.0000000) | where TIMESTAMP > ago (3d) | where Name == "PostgreSQL" | project TIMESTAMP, CPUPercent, MemPercent |render timechart

Session Stats:

MonOBPgSqlSessionsStatus | where LogicalServerName == "myflexserver" | where TIMESTAMP >= ago(2h)

Logs:

MonPgLogs | where LogicalServerName == "ServerName" |where TIMESTAMP >= starttime and TIMESTAMP <= Endtime |project TIMESTAMP, LogicalServerName, MachineName, ServerLocation,functionname,message_id |order by TIMESTAMP desc

How good have you found this content?

