Troubleshoot PG Flexible Server was not available

Last updated by | Hamza Aqel | Mar 31, 2023 at 6:05 AM PDT

For any changes please refer to haaqel@microsoft.com

Contents

On this TSG we are talking on how to troubleshoot and find the root cause of our high availability RCA cases with some canned RCAs around this, you should walk through these scenarios to see which of them related to your case:

Important:

Before moving forward with this TSG, you have to run the availability analyzer as explained in this TSG Availability Analyzer in ASC.

General spot checks

Availability Metric:

Availability can be measured using a metric. Each point represents 12 seconds of availability - if a point is missing, the server might've been unavailable in that 12s window. The metric can be seen using the below query function (with the right timestamps and server name):

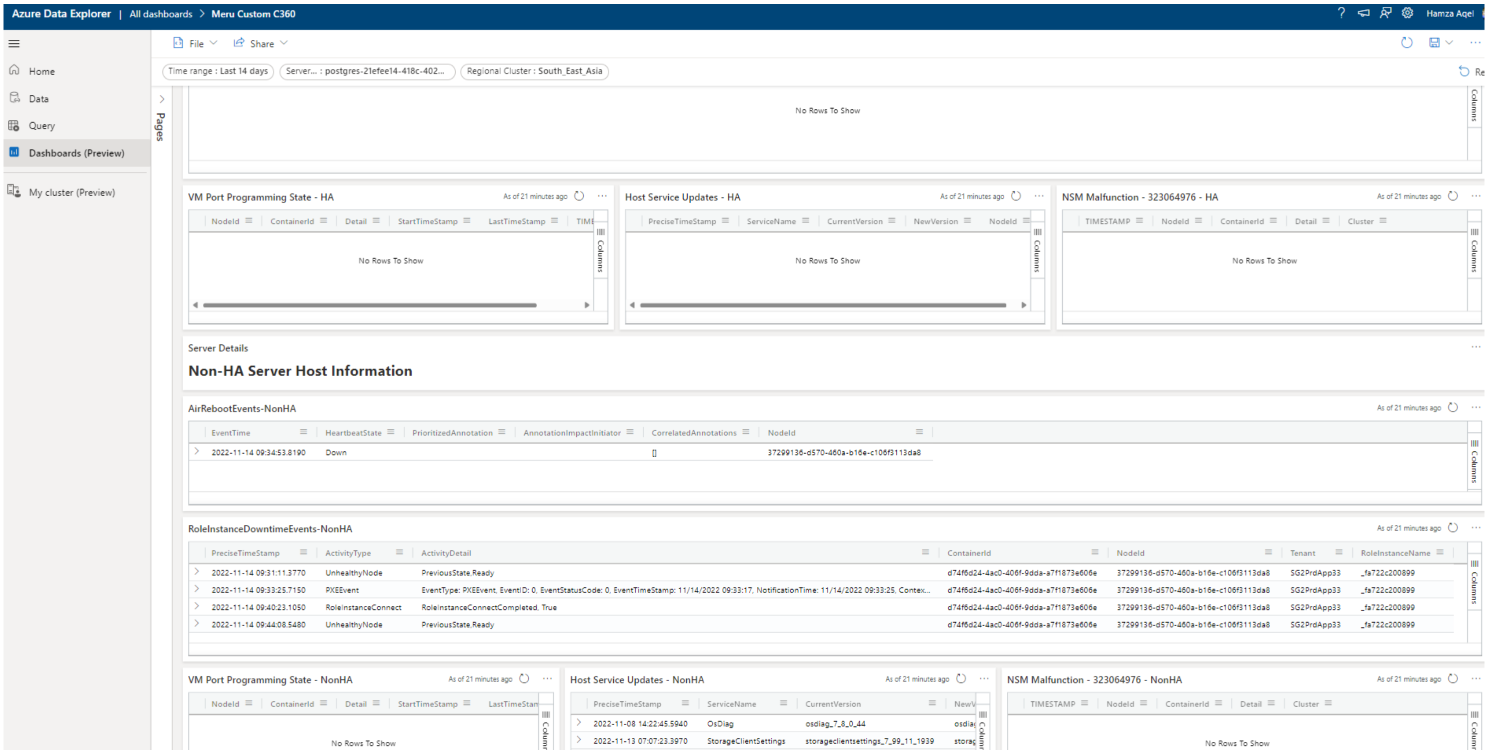
```
GetFSPGAvailabilityByServerTimeInterval(startTime=datetime(2022-09-12 09:00:00), endTime=datetime(2022-09-15 09:00:00), serverName="pgflexservername", granularity=tim
| where ReadAvailabilityPoints < ExpectedAvailabilityPoints
```

The query is very expensive, so the timestamps will need to be as close as possible (1 day at a time is acceptable). The function generates the metric from a combination of different logs. This query is not the only way to know whether a server was unavailable, but it is the easiest. After you know the time, you can use it in the below and coming queries.

Timestamp	Region	ServerType	OriginalPrimaryServerName	ReplicationSetId	HAEnabled	ServerEdition	ServerSku	VCores	SubscriptionId	SubscriptionType	TotalNumberofDataPo
2022-11-21 16:38:00.0000000	westeurope	PostgreSQL	pgflexservername	8D6606F9-0ADE-4D04-B60A-BFF64212C3BF	No	GeneralPurpose	Standard_D2s_v3	2	2FDEA10F-AC26-42EF-8147-4B77AE49F05C	Internal	5
2022-11-21 16:39:00.0000000	westeurope	PostgreSQL	pgflexservername	8D6606F9-0ADE-4D04-B60A-BFF64212C3BF	No	GeneralPurpose	Standard_D2s_v3	2	2FDEA10F-AC26-42EF-8147-4B77AE49F05C	Internal	5

Dashboard:

You can also use this dashboard to FastTrack this stage of investigation(requires 'RDOS Kusto Viewers' SG membership in idweb).



Time range : Last 14 days
Server... : postgres-21efee14-418c-402...
Regional Cluster : South_East_Asia

No Rows To Show

VM Port Programming State - HA As of 21 minutes ago

NodeId	ContainerId	Detail	StartTimeStamp	LastTimeStamp	TIME

No Rows To Show

Host Service Updates - HA As of 21 minutes ago

PreciseTimeStamp	ServiceName	CurrentVersion	NewVersion	NodeId

No Rows To Show

NSM Malfunction - 323064976 - HA As of 21 minutes ago

TIMESTAMP	NodeId	ContainerId	Detail	Cluster

No Rows To Show

Server Details

Non-HA Server Host Information

AirRebootEvents-NonHA As of 21 minutes ago

EventTime	HeartbeatState	PrioritizedAnnotation	AnnotationImpactInitiator	CorrelatedAnnotations	NodeId
2022-11-14 09:34:53.8190	Down			[]	37299136-d570-460a-b16e-c106f3113da8

RoleInstanceDowntimeEvents-NonHA As of 21 minutes ago

PreciseTimeStamp	ActivityType	ActivityDetail	ContainerId	NodeId	Tenant	RoleInstanceName
2022-11-14 09:31:11.3770	UnhealthyNode	PreviousState,Ready	d74f6d24-4ac0-406f-9dda-a7f1873e606e	37299136-d570-460a-b16e-c106f3113da8	SG2PrdApp33	_fa722c200899
2022-11-14 09:33:25.7150	PXEEvent	EventType: PXEEvent, EventID: 0, EventStatusCode: 0, EventTimeStamp: 11/14/2022 09:33:17, NotificationTime: 11/14/2022 09:33:25, Contex...	d74f6d24-4ac0-406f-9dda-a7f1873e606e	37299136-d570-460a-b16e-c106f3113da8	SG2PrdApp33	_fa722c200899
2022-11-14 09:40:23.1050	RoleInstanceConnect	RoleInstanceConnectCompleted, True	d74f6d24-4ac0-406f-9dda-a7f1873e606e	37299136-d570-460a-b16e-c106f3113da8	SG2PrdApp33	_fa722c200899
2022-11-14 09:44:08.5480	UnhealthyNode	PreviousState,Ready	d74f6d24-4ac0-406f-9dda-a7f1873e606e	37299136-d570-460a-b16e-c106f3113da8	SG2PrdApp33	_fa722c200899

VM Port Programming State - NonHA As of 21 minutes ago

NodeId	ContainerId	Detail	StartTimeStamp	LastTimeStam

No Rows To Show

Host Service Updates - NonHA As of 21 minutes ago

PreciseTimeStamp	ServiceName	CurrentVersion	NewV
2022-11-08 14:22:45.5940	OsDiag	osdiag_7_8_0_44	osdiag
2022-11-13 07:07:23.3970	StorageClientSettings	storageclientsettings_7_99_11_1939	storag

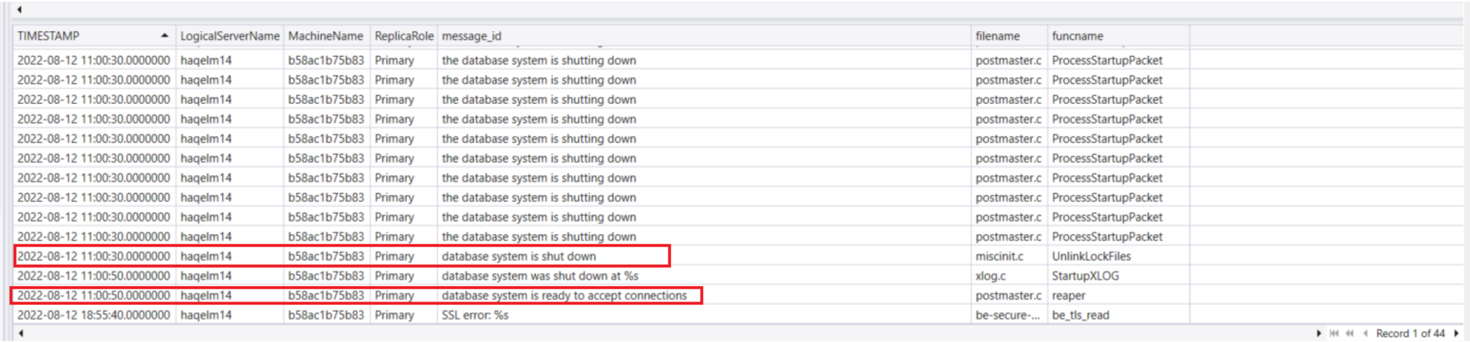
NSM Malfunction - 323064976 - NonHA As of 21 minutes ago

TIMESTAMP	NodeId	ContainerId	Detail	Cluster

No Rows To Show

-MonPGLogs:

```
MonPgLogs

| where TIMESTAMP >= datetime(2022-08-12 11:00:20.0000000) and TIMESTAMP <= datetime(2022-08-13 12:00:00.0000000)

| where LogicalServerName == "pgflexservername"

| where message_id has "recovery mode"

or message_id has "database system"

or message_id contains "fatal"

or message_id contains "starting up"

or message_id contains "error"

or message_id contains "downloading"

or message_id contains "terminat"

or message_id contains "ready to accept"

| project TIMESTAMP,LogicalServerName,MachineName,ReplicaRole, message_id, filename, funcname
```



TIMESTAMP	LogicalServerName	MachineName	ReplicaRole	message_id	filename	funcname
2022-08-12 11:00:30.0000000	haqelm14	b58ac1b75b83	Primary	the database system is shutting down	postmaster.c	ProcessStartupPacket
2022-08-12 11:00:30.0000000	haqelm14	b58ac1b75b83	Primary	the database system is shutting down	postmaster.c	ProcessStartupPacket
2022-08-12 11:00:30.0000000	haqelm14	b58ac1b75b83	Primary	the database system is shutting down	postmaster.c	ProcessStartupPacket
2022-08-12 11:00:30.0000000	haqelm14	b58ac1b75b83	Primary	the database system is shutting down	postmaster.c	ProcessStartupPacket
2022-08-12 11:00:30.0000000	haqelm14	b58ac1b75b83	Primary	the database system is shutting down	postmaster.c	ProcessStartupPacket
2022-08-12 11:00:30.0000000	haqelm14	b58ac1b75b83	Primary	the database system is shutting down	postmaster.c	ProcessStartupPacket
2022-08-12 11:00:30.0000000	haqelm14	b58ac1b75b83	Primary	the database system is shutting down	postmaster.c	ProcessStartupPacket
2022-08-12 11:00:30.0000000	haqelm14	b58ac1b75b83	Primary	the database system is shutting down	postmaster.c	ProcessStartupPacket
2022-08-12 11:00:30.0000000	haqelm14	b58ac1b75b83	Primary	the database system is shutting down	postmaster.c	ProcessStartupPacket
2022-08-12 11:00:30.0000000	haqelm14	b58ac1b75b83	Primary	the database system is shutting down	postmaster.c	ProcessStartupPacket
2022-08-12 11:00:30.0000000	haqelm14	b58ac1b75b83	Primary	database system is shut down	miscinit.c	UnlinkLockFiles
2022-08-12 11:00:50.0000000	haqelm14	b58ac1b75b83	Primary	database system was shut down at %s	xlog.c	StartupXLOG
2022-08-12 11:00:50.0000000	haqelm14	b58ac1b75b83	Primary	database system is ready to accept connections	postmaster.c	reaper
2022-08-12 18:55:40.0000000	haqelm14	b58ac1b75b83	Primary	SSL error: %s	be-secure-...	be_tls_read

Record 1 of 44

-OBvmagentsidecarpgsql:

```
OBvmagentsidecarpgsql

| where TIMESTAMP >= datetime(2022-08-26 10:00:00.0000000) and TIMESTAMP <= datetime(2022-08-26 11:00:00.0000000)

| where LogicalServerName == "haqelm14"
```

| where MessageString !contains "GetPgBouncerHealthReport: Ping returned"

| where MessageString contains "error" or MessageString contains "fatal"

| project TIMESTAMP,MessageString

TIMESTAMP	MessageString
2022-08-26 10:48:40.0000000	[PostgreSqlDbEngineHealth].Ping: Get error message on ping connection 57P01: terminating connection due to administrator command
2022-08-26 10:48:50.0000000	[PostgreSqlDbEngineHealth].Ping: Get error message on ping connection The connection was previously broken because of the following exception
2022-08-26 10:48:50.0000000	[PostgresSyntheticTransaction].RunSyntheticTransaction:Ignoring Error [Npgsql.PostgresException (0x80004005): 57P03: the database system is shutting down at Npgsql.NpgsqlConnector.<ReadMessage>g__ReadMessageLong|194_0(NpgsqlConnector connector, Boolean async, DataRowLoadingMode dataRowLoadingMode, Boolean readingNotifications, Boolean isReadingPrependedMessage) at Npgsql.NpgsqlConnector.Authenticate(String username, NpgsqlTimeout timeout, Boolean async, CancellationToken cancellationToken) at Npgsql.NpgsqlConnector.Open(NpgsqlTimeout timeout, Boolean async, CancellationToken cancellationToken) at Npgsql.ConnectorPool.OpenNewConnector(NpgsqlConnection conn, NpgsqlTimeout timeout, Boolean async, CancellationToken cancellationToken) at Npgsql.ConnectorPool.<>c__DisplayClass38_0.<<Rent>g__RentAsync|0>d.MoveNext() --- End of stack trace from previous location where exception was thrown --- at Npgsql.NpgsqlConnection.<>c__DisplayClass41_0.<<Open>g__OpenAsync|0>d.MoveNext()
2022-08-26 10:48:40.0000000	System.AggregateException: One or more errors occurred. (57P01: terminating connection due to administrator command) ---> Npgsql.PostgresException (0x80004005): 57P01: terminating connection due to administrator command at Npgsql.NpgsqlConnector.<ReadMessage>g__ReadMessageLong|194_0(NpgsqlConnector connector, Boolean async, DataRowLoadingMode dataRowLoadingMode, Boolean readingNotifications, Boolean isReadingPrependedMessage) at Npgsql.NpgsqlDataReader.NextResult(Boolean async, Boolean isConsuming, CancellationToken cancellationToken) at Npgsql.NpgsqlDataReader.NextResult() at Npgsql.NpgsqlCommand.ExecuteReader(CommandBehavior behavior, Boolean async, CancellationToken cancellationToken) at Npgsql.NpgsqlCommand.ExecuteReader(CommandBehavior behavior, Boolean async, CancellationToken cancellationToken)

Note:

You can remove the condition above and walk through the logs for more deeper analysis if there were no errors or nothing related.

- MonOBDockerContainerEvents

Check if there is any unhealth container at a certain time:

MonOBDockerContainerEvents

| where TIMESTAMP >= datetime(2022-08-26 10:00:00.0000000) and TIMESTAMP <= datetime(2022-08-26 11:00:00.0000000)

| where LogicalServerName == 'haqelm14'

| where Event contains "unhealthy"

| project TIMESTAMP,Event,atContainer,image,ServerVersion,AppVersion,AppType

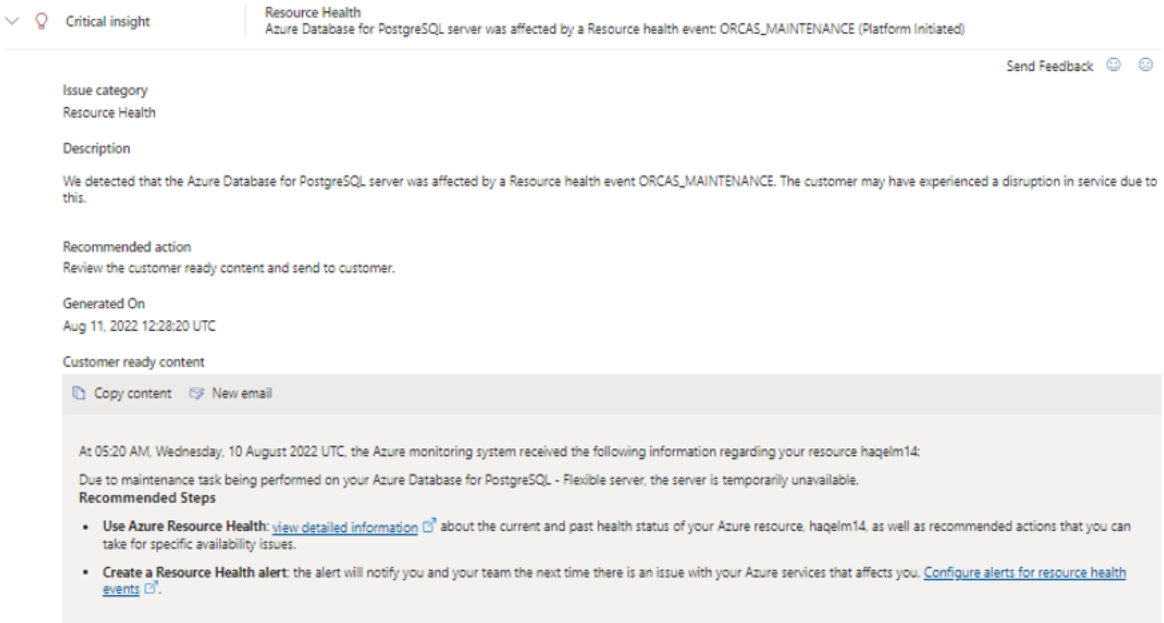
----------> > >

Now let's walk through some possible scenarios, take into consideration that we are not trying to list all the scenarios rather than to help in how to start with such cases:

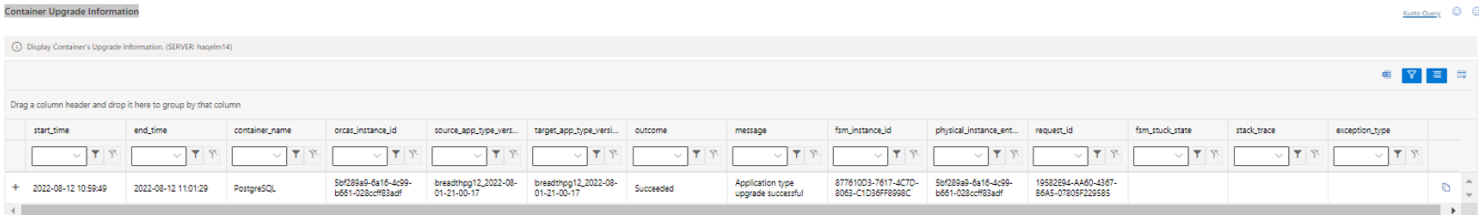
Check for any planned maintenance

For any case reported, the first thing to think about is if the server was under planned maintenance or not, we have wide range of possible ways in how to check and see if that matches the customer timestamp, either by using ASC through the ASC insights,..etc, or through Kusto/XTS

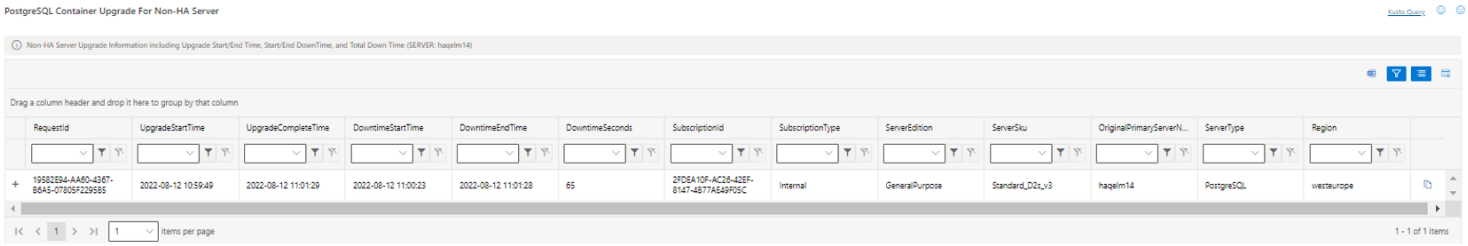
A- ASC:



You can check tab connectivity → Container Upgrade Information:

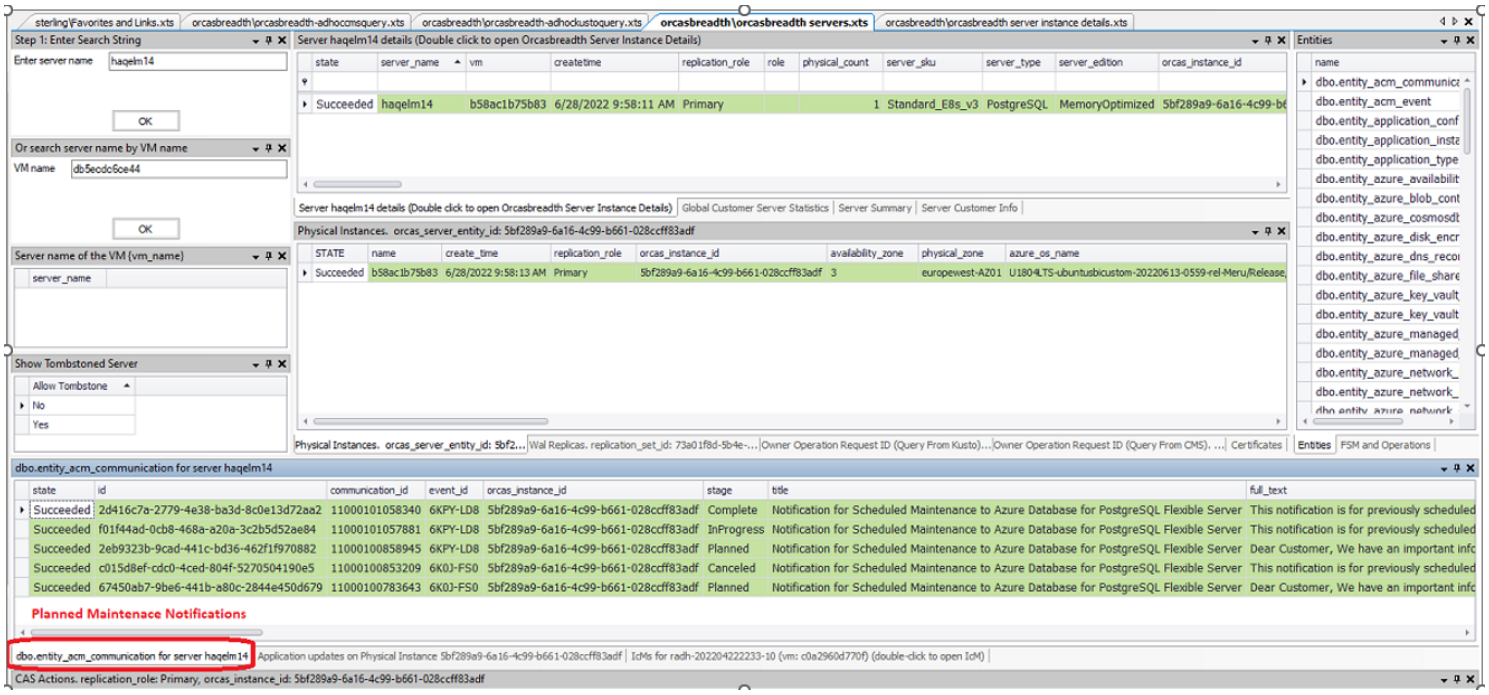


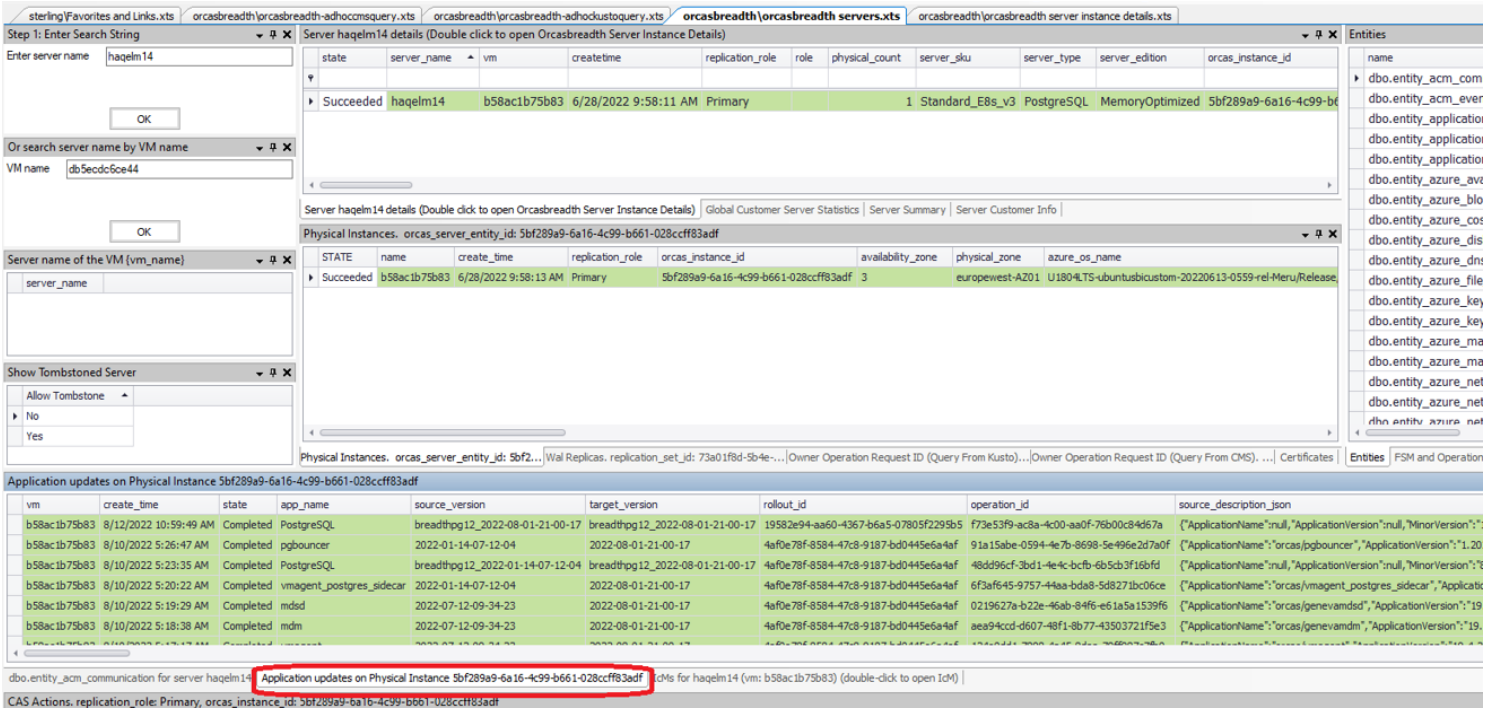
Customer Maintenance → PostgreSQL Container Upgrade for Non-HA /HA server:



B- XTS:

You can use this XTS view “orcasbreadth\orcasbreadth servers.xts” to explore all of these details:





C- Useful Kusto Queries for container upgrades:

Check containers Upgrade:

```
let STARTTIME_ARG = datetime('2022-08-11 13:59:00');

        let ENDTIME_ARG = datetime('2022-08-14 13:59:00');

        let SERVERNAME_ARG = 'haqelm14';

        let extract_exception_message = (message:string)

        {

            let index1 = indexof(message, 'ExceptionMessage');

            let index2 = indexof(message, 'ExceptionDetails');

            iif(index1 == -1, message, substring(message, index1, index2 - index1 - 6))

        };

        let extract_retry_message = (message:string)

        {

            let index1 = indexof(message, 'Retry 4');

            iif(index1 == -1, message, substring(message, 0, index1))

        };

        MonOrcasBreadthRp

        | where originalEventTimestamp > STARTTIME_ARG  and originalEventTimestamp < ENDTIME_ARG

        | where logical_server_name == SERVERNAME_ARG

        | where event =='server_app_type_upgrade_start_event'

        | project originalEventTimestamp, request_id, fsm_instance_id, logical_server_name, orcas_instance_id, physical_instance_entity_id, source_app_type, source_app_type_version,
target_app_type_version

        | join  kind = leftouter  (

            MonOrcasBreadthRp

            | where

                event == 'server_app_type_upgrade_success_event'

                or event == 'server_app_type_upgrade_failed_event'

                or (event in ('fsm_executed_action_failed','orcasbreadth_fsm_exception') and state_machine_type == 'ApplicationInstanceUpdateOperation')
```

```
        | extend  message = iff(isnotempty(message), message, upgrade_message)

        | summarize arg_max(end_time=originalEventTimestamp, event, state, exception_type, message, stack_trace) by fsm_instance_id

        | extend outcome = iif(event == 'server_app_type_upgrade_success_event', 'Succeeded',iif(event == 'server_app_type_upgrade_failed_event', 'Failed','Timeout'))

    ) on fsm_instance_id

    | extend message = extract_exception_message(message)

    | extend message = extract_retry_message(message)

    | project start_time=originalEventTimestamp, end_time, container_name=source_app_type, orcas_instance_id, source_app_type_version, target_app_type_version, outcome, message, fsm_instance_id, physical_instance_entity_id, request_id, fsm_stuck_state=state, stack_trace, exception_type

    | order by start_time desc
```

start_time	end_time	container_name	orcas_instance_id	source_app_type_version	target_app_type_version	outcome	message	fsm_instance_id	physical_instance_entity_id	request_id
2022-08-12 10:59:49.7493899	2022-08-12 11:01:29.9668205	PostgreSQL	5bf289a9-6a16-4c99-b661-028ccff83adf	breadthpg12_2022-08-01-21-00-17	breadthpg12_2022-08-01-21-00-17	Succeeded	Application type upgrade successful	877610D3-7617-4C7D-8063-C1D36FF8998C	5bf289a9-6a16-4c99-b661-028ccff83adf	19582E94-AA60-4367 B6A5-07805F229

```
let SERVERNAME_ARG = 'haqelm14';

    let STARTTIME_ARG = datetime('2022-08-11 14:01:00');

    let ENDTIME_ARG = datetime('2022-08-14 14:01:00');

    let GetFSPGDbContainerUpgradesNonHA = (startTime:datetime, endTime:datetime){

    MonOrcasBreadthRp

    | where PreciseTimeStamp > startTime and PreciseTimeStamp <= endTime and AppName == 'OrcasBreadthRp'

    | where event in ('server_app_type_upgrade_success_event', 'server_app_type_upgrade_failed_event') and source_app_type == 'PostgreSQL'

    | extend Region = GetRegionNameFromFabricCluster(ClusterName)

    | project logical_server_name, orcas_instance_id, source_app_type, source_app_type_version, target_app_type_version, fsm_instance_id, Region, event, request_id, UpgradeCompleteTime = originalEventTimestamp

    | join kind = inner (

        MonOrcasBreadthRp

        | where event == 'server_app_type_upgrade_start_event' and source_app_type == 'PostgreSQL'

        | project fsm_instance_id, timebin = bin(PreciseTimeStamp, 30m), UpgradeStartTime = originalEventTimestamp

        | extend prevbin = datetime_add('minute', -30, timebin)

    ) on fsm_instance_id

    | join kind = inner (

        MonOBDirectorV2ActorEvents

        | where isnotempty( OriginalPrimaryServerName) and ReplicationRole == 'Primary' and EngineName == 'PostgreSQL'

        | summarize by ServerName, OriginalPrimaryServerName, EngineName, OrcasInstanceId, timebin = bin(PreciseTimeStamp, 30m), ReplicationSetId

    ) on $left.logical_server_name == $right.ServerName and $left.prevbin == $right.timebin

    | where isempty( orcas_instance_id) or orcas_instance_id =~ OrcasInstanceId

    | join kind = inner (

        MonBillingMeruSinglePgServerStatus

        | extend ha_enabled = iff(column_ifexists('ha_standby_servers_count', 0) > 0, 1, 0)

        | summarize by server_edition, server_sku, subscription_id, server_name, snapshotbin = bin(PreciseTimeStamp, 30m), ha_enabled

    ) on $left.OriginalPrimaryServerName == $right.server_name and $left.prevbin == $right.snapshotbin

    | where ha_enabled == 0

    | join kind = leftouter (

        MonOrcasBreadthRp

        | where state_machine_type == 'ApplicationInstanceUpdateOperation' and state == 'DroppingOldContainer'
```

```
    | summarize upTimeStamp = min(originalEventTimestamp) by fsm_instance_id
) on fsm_instance_id
| join kind = leftouter (
    MonOrcasBreadthRp
    | where state_machine_type == 'ApplicationInstanceUpdateOperation' and state in ('StoppingOldContainer', 'GracefulKillOldContainer')
    | summarize downTimeStamp = max(originalEventTimestamp) by fsm_instance_id
) on fsm_instance_id
| extend DowntimeSeconds = round(iff(event == 'server_app_type_upgrade_success_event', totimespan(upTimeStamp - downTimeStamp) / time(1s),real(0)))
| extend subscription_type = iff(subscription_id in~ (GetInternalSubscriptions()), 'Internal', 'External')
| project RequestId = request_id,
        UpgradeStartTime,
        UpgradeCompleteTime,
        DowntimeStartTime = downTimeStamp,
        DowntimeEndTime = upTimeStamp,
        DowntimeSeconds,
        SubscriptionId = subscription_id,
        SubscriptionType = subscription_type,
        ServerEdition = server_edition,
        ServerSku = server_sku,
        OriginalPrimaryServerName,
        ServerType = EngineName,
        Region = Region
};
GetFSPGDbContainerUpgradesNonHA(STARTTIME_ARG, ENDTIME_ARG)
| where OriginalPrimaryServerName =~ SERVERNAME_ARG and ServerType == 'PostgreSQL'
```

RequestId	UpgradeStartTime	UpgradeCompleteTime	DowntimeStartTime	DowntimeEndTime	DowntimeSeconds	SubscriptionId	SubscriptionType	ServerEdition	ServerSku	OriginalPrima
19582E94-AA60-4367-B6A5-07805F2295B5	2022-08-12 10:59:49.7493899	2022-08-12 11:01:29.9668205	2022-08-12 11:00:23.6077395	2022-08-12 11:01:28.4862931	65	2FDEA10F-AC26-42EF-8147-4B77AE49F05C	Internal	GeneralPurpose	Standard_D2s_v3	haqelm14

Root Cause:

Your Azure database for PostgreSQL Flexible server was under planned maintenance between DowntimeStartTime and DowntimeEndTime . You will experience a restart of the server during the maintenance window where we perform periodic maintenance to keep your managed database secure, stable, and up-to-date. During maintenance, the server gets new features, updates, and patches.

References:

Learn more about Planned maintenance in Azure Database for Flexible Servers.

- Learn how to change the maintenance schedule
- Learn how to get notifications about upcoming maintenance using Azure Service Health
- Learn how to set up alerts about upcoming scheduled maintenance events

Check If any server restart occurred:

Check if there is any restart operation and correlate with the server availability time.

MonOrcasBreadthRp

```
| where TIMESTAMP >= ago(3d)

| where operation_type == "RestartServerManagementOperation"

| where operation_parameters contains "pgflexservername"

| project TIMESTAMP,request_id,event,operation_parameters
```

TIMESTAMP	request_id	event	operation_parameters
2022-08-26 10:48:43.0342312	B63808F2-72CF-4D80-8D1C-C1BDA334FDC0	management_operation_start	<?xml version="1.0"?> <InternalServerPowerOperationArguments xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <SubscriptionId>2fdea10f-ac26-42ef-8147-4b77ae49f05c</SubscriptionId> <ResourceGroupName>haqelnegrp</ResourceGroupName> <ServerName>haqelm14</ServerName> <ApplicationName>PostgreSQL</ApplicationName> <PowerOperationType>Restart</PowerOperationType> <ServerEntityId>00000000-0000-0000-0000-000000000000</ServerEntityId>
2022-08-26 10:49:13.0350740	B63808F2-72CF-4D80-8D1C-C1BDA334FDC0	management_operation_success	<?xml version="1.0"?> <InternalServerPowerOperationArguments xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <SubscriptionId>2fdea10f-ac26-42ef-8147-4b77ae49f05c</SubscriptionId> <ResourceGroupName>haqelnegrp</ResourceGroupName> <ServerName>haqelm14</ServerName> <ApplicationName>PostgreSQL</ApplicationName> <PowerOperationType>Restart</PowerOperationType> <ServerEntityId>5bf289a9-6a16-4c99-b661-028ccff83adf</ServerEntityId>

Check if there any VM node issue

Get the role instance name:

```
MonPgLogs

| where TIMESTAMP >= ago(1h)

| where LogicalServerName == "flexpgbouncer1"

| distinct RoleInstance,ReplicaRole
```

RoleInstance	ReplicaRole
prod:westeurope:a4545bb80d6f	Standby
prod:westeurope:a9f10f814abd	Primary

You will have two records in case HA is enabled, and 1 record in case of non-HA servers. You need to check both instances:

Check for node/VM events:

```
cluster('vmainsight.kusto.windows.net').database('Air'). AirRebootEvents


| where EventTime > ago(30d)

| where RoleInstanceName  == "_ a4545bb80d6f " or RoleInstanceName  == "_ a9f10f814abd "

| project EventTime, NodeId, PrioritizedAnnotation, AnnotationType, RCALevel1, RCALevel2, RCALevel3

| order by EventTime
```

EventTime	NodeId	PrioritizedAnnotation	AnnotationType	RCALevel1	RCALevel2	RCALevel3
2022-05-25 04:32:52.9880000	37340683-8165-b876-a32a-abe698deefa2	VirtualMachineRestarted	DowntimeUnexpected	VirtualDiskFault	ToRDown_Networking	[PhynetDiag]AdditionalRCA: ToRDeviceUnhealthy_Networking;ToRDown_Networking;TorSuspicious_Netwo AllUnhealthySignals: d_check_is_device_healthy_kusto|SnmpLoopbackReachab DeviceName: mnz22-0101-1101-07t0, OsVersion: SONiC.20181130.95, Hardwa 7060CX-32S-D48C8, DeviceLifeCycle: InProduction, ReloadCause: .
2022-05-25 04:31:46.1380000	37340683-8165-b876-a32a-abe698deefa2	VirtualMachineRestarted	DowntimeUnexpected	VirtualDiskFault	ToRDown_Networking	[PhynetDiag]AdditionalRCA: ToRDeviceUnhealthy_Networking;ToRDown_Networking;TorSuspicious_Netwo AllUnhealthySignals: d_check_is_device_healthy_kusto|SnmpLoopbackReachab DeviceName: mnz22-0101-1101-07t0, OsVersion: SONiC.20181130.95, Hardwa 7060CX-32S-D48C8, DeviceLifeCycle: InProduction, ReloadCause: .
2022-05-25 04:30:38.8460000	37340683-8165-b876-a32a-abe698deefa2	VirtualMachineRestarted	DowntimeUnexpected	NodeFault	UnhealthyNode_Faulty NIC or Driver not loaded	
2022-05-25 04:29:41.5760000	37340683-8165-b876-a32a-abe698deefa2	VirtualMachineRestarted	DowntimeUnexpected	NodeFault	UnhealthyNode_Faulty NIC or Driver not loaded	
2022-05-25 04:28:40.9240000	37340683-8165-b876-a32a-abe698deefa2	VirtualMachineStorageOffline	DowntimeUnexpected	NodeFault	UnhealthyNode_Faulty NIC or Driver not loaded	
2022-05-25 04:27:33.9000000	37340683-8165-b876-a32a-abe698deefa2	VirtualMachineStorageOffline	DowntimeUnexpected	NodeFault	UnhealthyNode_Faulty NIC or Driver not loaded	

That Kusto table shows you any hardware or VM related operations, with RCAs starting from high level one (RCALevel1) and more details through RCALevel2 till RCA Level3, we can write an RCA based on these columns (you can consult with your EEE if you have doubts or need a help with that):

Root Cause:

PG Servers:
pgflexservername was not available at yyyy-mm-dd hh24:mi:ss

DESCRIPTION:
At around yyyy-mm-dd hh24:mi:ss the server experienced unavailability for about x mins.

ROOT CAUSE:
Due to a network issue on Azure platform, the Virtual Machine of the primary node that hosts PostgreSQL's briefly lost connection to its remote disk, which caused unplanned downtime and a filesystem problem on the OS. In this case, the VM went down and auto recovered, we apologize for the inconvenience caused by this unavailability. We apologize for the inconvenience caused by this incident.

In some cases, there are not much info in that table, for example:

Execute: [Web] [Desktop] [Web (Lens)] [Desktop (SAW)] https://vmainsight.kusto.windows.net/Air

```
AirRebootEvents

| where RoleInstanceName == "_e10e5a06c36c"

| where EventTime between (datetime(2022-07-12).. 1d)

| project EventTime, NodeId, PrioritizedAnnotation, AnnotationType, RCALevel1, RCALevel2, RCALevel3
```

EventTime	NodeId	PrioritizedAnnotation	AnnotationType	RCALevel1	RCALevel2
2022-07-12 16:57:54.9660000	6966ca6a-ac9e-205a-7176-a1b3113aa4aa	VirtualMachineMigrationInitiatedForRepair	DowntimeUnexpected	NodeFault	UnhealthyNode_Inconclusive: OS Liveness Undetermined

You can extract more info using the below query:

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

```
LogContainerHealthSnapshot

| where roleInstanceName=='_e10e5a06c36c'

| where PreciseTimeStamp  between (datetime(2022-07-12).. 1d)

| summarize min(PreciseTimeStamp), max(PreciseTimeStamp) by roleInstanceName, Tenant, nodeId, containerId, containerLifecycleState, containerIsolationState, actualOperationalState, faultInfo
```

roleInstanceName	Tenant	nodeId	containerId	containerLifecycleState	containerIsolationState	actualOperationalState	min_PreciseTimeStamp	max_PreciseTimeStamp
_e10e5a06c36c	MNZ22PrdApp37	6966ca6a-ac9e-205a-7176-a1b3113aa4aa	c82f2723-92cc-4979-b6ef-64f25667993e	Alive	NotIsolated	Up	2022-07-12 00:11:21.9254819	2022-07-12 16:50:31.8557901
_e10e5a06c36c	MNZ22PrdApp37	6966ca6a-ac9e-205a-7176-a1b3113aa4aa	c82f2723-92cc-4979-b6ef-64f25667993e	Alive	NotIsolated	Unknown	2022-07-12 09:51:43.2100319	2022-07-12 17:05:20.9447976
_e10e5a06c36c	MNZ22PrdApp32	0b5db639-fe68-ce08-1172-97cc46370077	ec63632d-9c42-4363-abc6-e439200ffa5b	Alive	NotIsolated	Down	2022-07-12 17:05:20.7487218	2022-07-12 17:05:57.6310773
_e10e5a06c36c	MNZ22PrdApp37	6966ca6a-ac9e-205a-7176-a1b3113aa4aa	c82f2723-92cc-4979-b6ef-64f25667993e	ToBeDestroyedOnNode	NotIsolated	Unknown	2022-07-12 17:05:20.9452562	2022-07-12 17:05:20.9452562
_e10e5a06c36c	MNZ22PrdApp37	6966ca6a-ac9e-205a-7176-a1b3113aa4aa	c82f2723-92cc-4979-b6ef-64f25667993e	ToBeDestroyedOnNode	NodeIsIsolated	Unknown	2022-07-12 17:05:33.6246963	2022-07-12 17:05:33.6246963
_e10e5a06c36c	MNZ22PrdApp37	6966ca6a-ac9e-205a-7176-a1b3113aa4aa	c82f2723-92cc-4979-b6ef-64f25667993e	Destroyed	NodeIsIsolated	Unknown	2022-07-12 17:05:33.6434047	2022-07-12 17:05:33.6434047
_e10e5a06c36c	MNZ22PrdApp32	0b5db639-fe68-ce08-1172-97cc46370077	ec63632d-9c42-4363-abc6-e439200ffa5b	Alive	NotIsolated	Up	2022-07-12 17:05:57.6569904	2022-07-12 23:55:01.1643553

And in this case as we can see the issue in the node, and you can use this RCA:

PG Servers:
pgflexservername was not available at yyyy-mm-dd hh24:mi:ss

DESCRIPTION:
At around yyyy-mm-dd hh24:mi:ss the server experienced unavailability for about x mins.

ROOT CAUSE:
At around yyyy-mm-dd hh24:mi:ss, the Azure host node hosting the VM for the PostgreSQL service faulted causing immediate loss of the service. The Azure compute platform relocated the VM to a healthy node and the service was restored around yyyy-mm-dd hh24:mi: ss. We apologize for the inconvenience caused by this unavailability.

Another sample result from AirRebootEvents:

EventTime	NodeId	PrioritizedAnnotation	AnnotationType	RCALevel1	RCALevel2	RCALevel3
2022-06-07 07:57:27.6880000	8a705ad0-41a5-87f5-569a-d53384826b6a	VirtualMachineDeallocationInitiated	DowntimeExpected	ContainerOperation	ContainerCreated CreateTenant,True,	CreateTenant,True,
2022-06-07 03:45:47.3790000	18a8bad1-704c-964d-674f-52e739d401c6			HostOSCrash	CORRUPT_MODULELIST_AV	->CORRUPT_MODULELIST_AV

You can write this RCA for example:

PG Servers:
pgflexservername was not available at yyyy-mm-dd hh24:mi:ss

DESCRIPTION:
At around yyyy-mm-dd hh24:mi:ss the server experienced unavailability for about x mins.

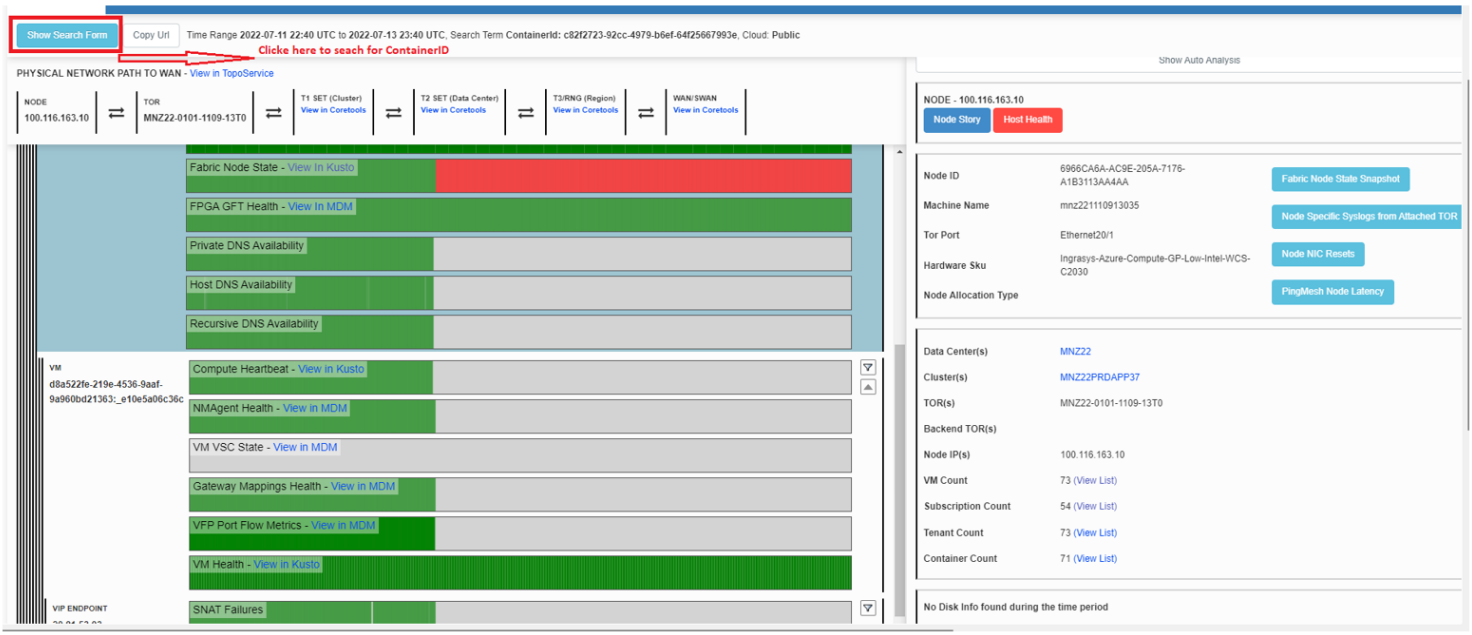
ROOT CAUSE:
At around yyyy-mm-dd hh24:mi:ss, Host crash caused OS disk cache to be lost, which caused the container to be cleaned up on docker start-up due to a corrupted config file.

Note:

You might see others related to Guest OS upgrade, fix ..etc and so on, and you can use the same way in writing the RCA, if you have any doubts, you can reach out to your EEE for help on RCAs if needed.

Check customer resource and node health

You can use this tool https://netvma.azure.net/?startTime=07%2F11%2F2022+22%3A40&endTime=07%2F13%2F2022+23%3A40&value=c82f2723-92cc-4979-b6ef-64f25667993e&destValue=&pathQuery=false&sdnPath=false



To get the ContainerID , you can use previously mentioned query :

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

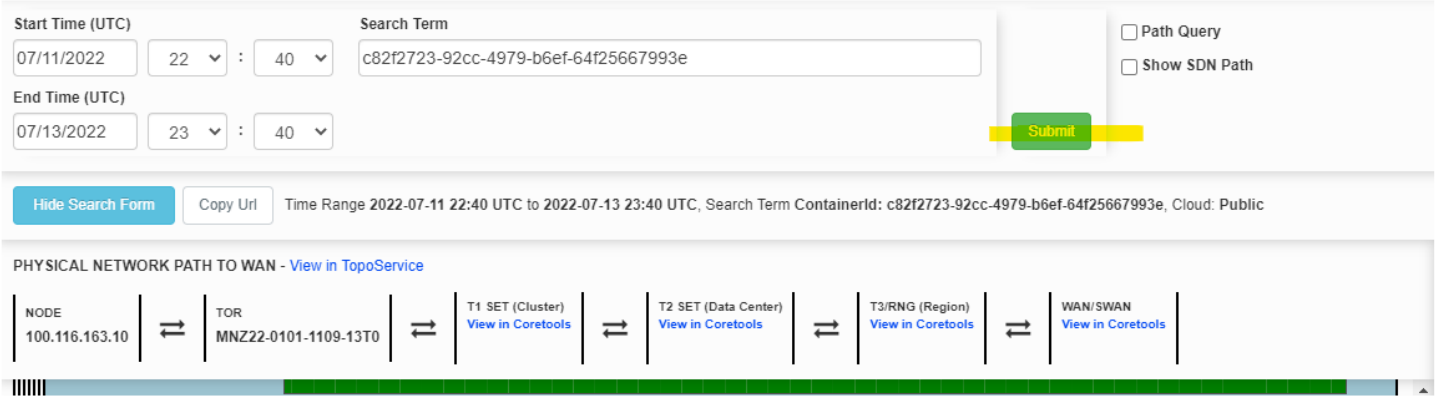
| where roleInstanceName=='_e10e5a06c36c'

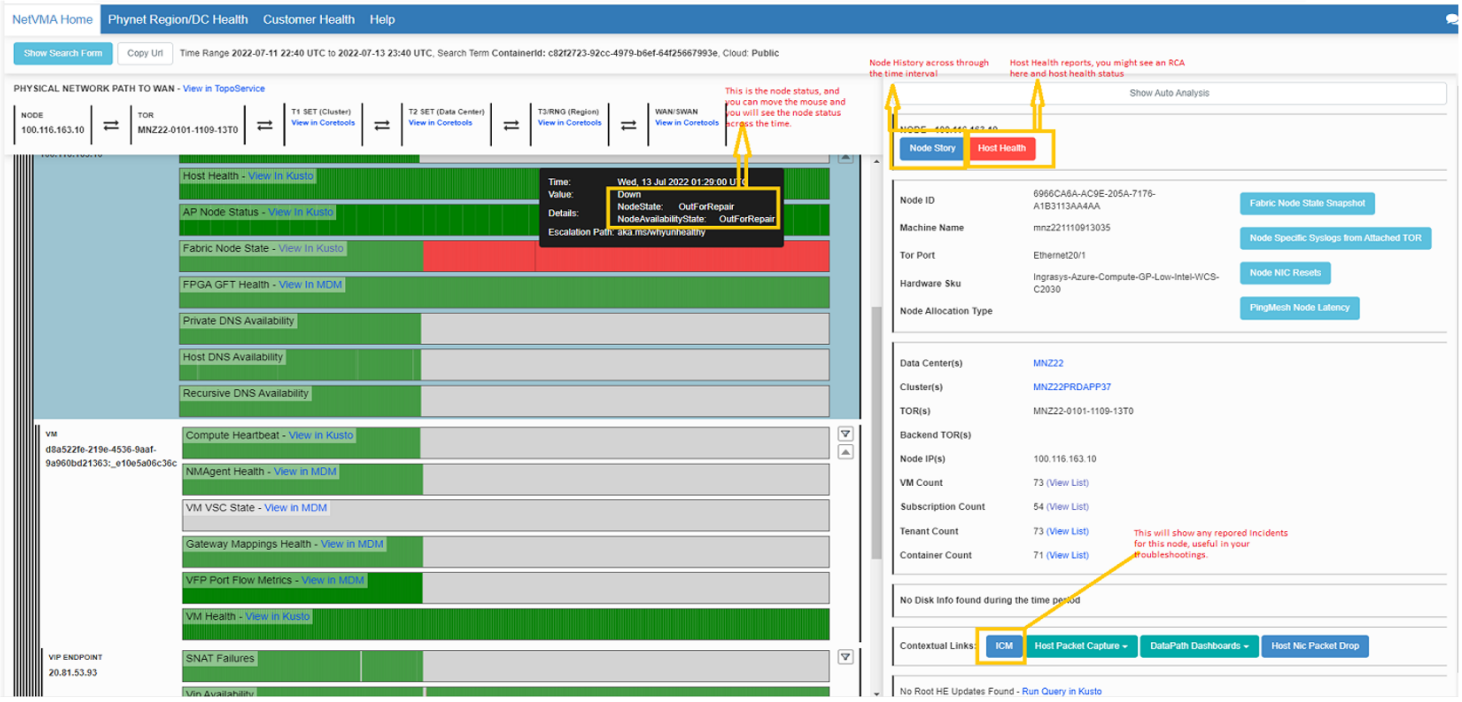
| where PreciseTimeStamp between (datetime(2022-07-12).. 1d)

| summarize min(PreciseTimeStamp), max(PreciseTimeStamp) by roleInstanceName, Tenant, nodeId, containerId, containerLifecycleState, containerIsolationState ,actualOperationalState, faultInfo

roleInstanceName	Tenant	nodeId	containerId	containerLifecycleState	containerIsolationState	actualOperationalState	min_PreciseTimeStamp	max_PreciseTimeStamp
_e10e5a06c36c	MNZ22PrdApp37	6966ca6a-ac9e-205a-7176-a1b3113aa4aa	c82f2723-92cc-4979-b6ef-64f25667993e	Alive	NotIsolated	Up	2022-07-12 00:11:21.9254819	2022-07-12 16:50:31.8557901
_e10e5a06c36c	MNZ22PrdApp37	6966ca6a-ac9e-205a-7176-a1b3113aa4aa	c82f2723-92cc-4979-b6ef-64f25667993e	Alive	NotIsolated	Unknown	2022-07-12 09:51:43.2100319	2022-07-12 17:05:20.9447976
_e10e5a06c36c	MNZ22PrdApp32	0b5db639-fe68-ce08-1172-97cc46370077	ec63632d-9c42-4363-abc6-e439200ffa5b	Alive	NotIsolated	Down	2022-07-12 17:05:20.7487218	2022-07-12 17:05:57.6310773
_e10e5a06c36c	MNZ22PrdApp37	6966ca6a-ac9e-205a-7176-a1b3113aa4aa	c82f2723-92cc-4979-b6ef-64f25667993e	ToBeDestroyedOnNode	NotIsolated	Unknown	2022-07-12 17:05:20.9452562	2022-07-12 17:05:20.9452562
_e10e5a06c36c	MNZ22PrdApp37	6966ca6a-ac9e-205a-7176-a1b3113aa4aa	c82f2723-92cc-4979-b6ef-64f25667993e	ToBeDestroyedOnNode	NodeIsIsolated	Unknown	2022-07-12 17:05:33.6246963	2022-07-12 17:05:33.6246963
_e10e5a06c36c	MNZ22PrdApp37	6966ca6a-ac9e-205a-7176-a1b3113aa4aa	c82f2723-92cc-4979-b6ef-64f25667993e	Destroyed	NodeIsIsolated	Unknown	2022-07-12 17:05:33.6434047	2022-07-12 17:05:33.6434047
_e10e5a06c36c	MNZ22PrdApp32	0b5db639-fe68-ce08-1172-97cc46370077	ec63632d-9c42-4363-abc6-e439200ffa5b	Alive	NotIsolated	Up	2022-07-12 17:05:57.6569904	2022-07-12 23:55:01.1643553

Use the ContainerID destroyed "c82f2723-92cc-4979-b6ef-64f25667993e"





Another way to investigate Platform Node Issue

There are multiple steps in identifying and confirming whether the node on which the VM is hosted had experienced a problem. We can check whether the node was unhealthy using the NodeEvents table.

We need to gather the VM NodeID, ContainerID using the below query:

```
cluster("Azurecm").database('AzureCM').LogContainerSnapshot
| where PreciseTimeStamp >= datetime(2022-08-14 12:24:00) and PreciseTimeStamp <= datetime(2022-08-14 14:24:00)
//| where subscriptionId contains "aa50eea1-b327-4730-9606-3edc083b8e83"  // MSFT subscription ID
| where roleInstanceName contains "dd2a3a6fec21"  // machine name
```

The timestamp for the above query needs to be a little wider since the snapshot is not taken all the time.. General idea is startTime = (issue time - 1h) and endTime = (issue time + 30m).

We can use the NodeID and timestamps from the previous query in the below query:

```
cluster("Azurecm").database('AzureCM').TMMgmtNodeEventsEtwTable
| where PreciseTimeStamp >= datetime(2022-08-26 15:17:32.000Z) and PreciseTimeStamp <= datetime(2022-08-26 17:17:32.000Z)
| where NodeId =~ "ee97324d-c157-a0fa-da7a-f7a321bf72a8"
```

We can filter for " `aka.ms` ", we can get the link to the Hawkeye dashboard that auto-RCAs node faults. This link in the logs is sufficient to indicate that there's a problem with the node. However, this is not enough to classify the problem and raise it to the right team upstream.

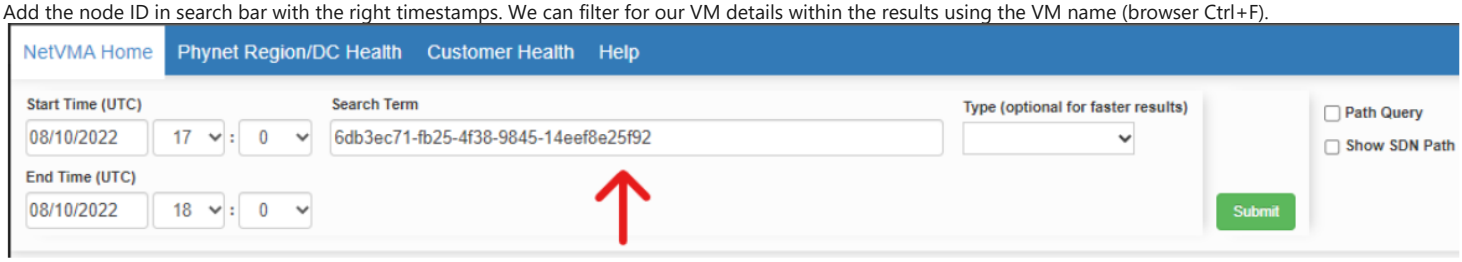
You can also use this dashboard to FastTrack this stage of investigation (requires 'RDOS Kusto Viewers' SG membership in idweb).

NetVMA

A good amount of networking information can be seen using NetVMA tool. It requires the nodeID for filtering, too. We can use the same query to fetch node ID:

```
cluster("Azurecm").database('AzureCM').LogContainerSnapshot
| where PreciseTimeStamp >= datetime(2022-08-14 12:24:00) and PreciseTimeStamp <= datetime(2022-08-14 14:24:00)
//| where subscriptionId contains "aa50eea1-b327-4730-9606-3edc083b8e83"  // MSFT subscription ID
| where roleInstanceName contains "dd2a3a6fec21"  // machine name
```

Add the node ID in search bar with the right timestamps. We can filter for our VM details within the results using the VM name (browser Ctrl+F).



NetVMA can be accessed using this link: https://netvma.azure.net/