Why did my ORCAS server restart?

Last updated by | Daniel Valero | Apr 8, 2022 at 10:11 AM PDT

Contents

How to identify a restart

NOTE Use one of the SandBox views depending on the product you are using

```
MonRdmsMySqlSandbox
MonRdmsPgSqlSandbox
| where LogicalServerName in ("<servername>")
| summarize min(originalEventTimestamp), max(originalEventTimestamp) by LogicalServerName, code_package_versio
```

Restart reasons and Troubleshooting steps

If the restart in question is sufficiently old (few days) all failovers with most causes categorized should be available in: https://avocado/reports/291362 (note that server names are hashed with SHA256)

When troubleshooting / RCA for an elastic server restart, please check the following common reasons for restart:

Crash

1. Check AzureWatson for dumps
2. Check sandbox logs for "crash" / "crashdump.exe" and or e.g. PANIC exceptions for PG.

Specific issue might be an existing problem with a TSG

Upgrade

1. Check the code_package_version change in sandbox logs as explained at [How to identify a restart]
2. Check XTS view for app type upgrades and recent infra upgrades. A few examples
   - sterling\sterlingupgradehistory.xts
   - sterling\infrawinfabstatus.xts

Update SLO

Identify a update SLO using any of the methods below:

- Check MonManagement telemetry, where operation parameters contains the server name
- Check the sandbox logs for AppName change together with process_id change
- Check *Analytics* snapshot in Kusto

Sample SLO query using sandbox log: (This query checks for appname change for a given logicalservername as the condition for SLO update event)

```
let TimeCheckStart = datetime('01/03/2019 00:31');
let TimeCheckEnd = datetime('01/04/2019 21:30');
MonRdmsPgSqlSandbox
| where originalEventTimestamp > TimeCheckStart and originalEventTimestamp < TimeCheckEnd
| where LogicalServerName  == 'elasticserver-pgsqlcrud-gabasic-eastus2-0d8b3b03' and ClusterName !contains 'ms
| summarize max(originalEventTimestamp) by LogicalServerName, AppName
| order by LogicalServerName asc, max_originalEventTimestamp asc
| summarize count(), makelist(max_originalEventTimestamp) by LogicalServerName
| where count_ > 1;
```

User initiated restart from Portal

```
MonManagementResourceProvider
| where request_url contains "{ServerName}"
| where request_url contains "restart"
```

or

```
AlrManagement
| where TIMESTAMP >= ago(4d) and elastic_server_name =="<server name>"
| where operation_type == 'RestartElasticServer'
| where event in ('management_operation_elastic_server')
| project originalEventTimestamp, ClusterName, operation_type, elastic_server_name, server_type, state, reques
```

Manual CAS restart

Outside of CSS scope

For example: Get-FabricNode -NodeName DB.3 -NodeClusterName tr210.eastus1-a.worker.database.windows.net | Kill-Process -ProcessName DkMon.exe -ApplicationNameUri fabric:/Worker.PAL.PG/blablabla

1. AudCAS in Diagnostics Prod Accessing CAS Audit logs (Web view)

2. Check NodeAgent events

```
MonNodeAgentEvents
| where TIMESTAMP >= datetime(2018-09-27 08:10:01.785) and TIMESTAMP <= datetime(2018-09-27 08:50:01.785)
| where NodeName == "DB.48" and ClusterName contains "tr23"

MonRdmsPgSqlSandbox
| where text contains "App20883"
| limit 100
```

Bot CAS restart

Check **sterling\bot actions.xts** XTS view

(at the time of writing, there is at least one PG BOT mitigating servers without heartbeat / with PICO process missing)

Planned failovers

```
MonNodeTraceETW
| where Message contains "DkMon:"
| where Message !contains "elasticserver"
| extend originalEventTimestampText  = replace('"','',extract('OriginalEventTimeStamp: "([0-9.]+)-([0-9.]+)-([
| extend milliseconds  = replace('"','',extract('OriginalEventTimeStamp: "([0-9.]+)-([0-9.]+)-([0-9.]+) ([0-9.
| extend ind = string_size(originalEventTimestampText)
| extend originalEventTimestamp = replace(':(?:[^:]+)$','',substring(originalEventTimestampText, 24, ind))
| extend originalEventTimestamp = todatetime(strcat(originalEventTimestamp, '.', milliseconds))
| extend EventMessage   = extract('Message: "(a-z|A-Z|0-9|.+)"', 1, Message, typeof(string))
| extend App = extract('fabric:/[a-z|A-Z|.]+/([a-z|A-Z|0-9]+)', 0, Message, typeof(string))
| extend AppName = extract('fabric:/[a-z|A-Z|.]+/([a-z|A-Z|0-9]+)', 1, Message, typeof(string))
| extend AppTypeName = iff(App contains 'Worker.PAL.MySQL', 'Worker.PAL.MySQL', 'Worker.PAL.PG')
| extend LogicalServerName   = extract('LogicalServerName: "(a-z|A-Z|0-9|.+).database.windows.net"', 1, Messag
| project originalEventTimestamp, process_id = Pid, NodeName, AppTypeName, AppName,LogicalServerName, EventMes
| distinct *
```

NOTE: If you see "DkMon: CTRL_C event received", this indicates planned restart by WinFab.

DevOps Node JIT

Security team should have an audit trail

Node restart

Good approximation is whether other applications on the node were restarted, too. Possible Kusto sources to check: the sandbox logs and WinFabLogs.

Use this to identify if all the servers on the Node restarted:

```
MonRdmsPgSqlSandbox
| where ClusterName == "tr311.westeurope1-a.worker.database.windows.net"
| where NodeName  == "DB.0"
| where originalEventTimestamp  <= datetime(2019-05-04 09:14:57.4059504) // failover time + few hours
| summarize max(originalEventTimestamp), min(originalEventTimestamp) by process_id , code_package_version, App
| extend upTime = max_originalEventTimestamp - min_originalEventTimestamp
| distinct LogicalServerName, max_originalEventTimestamp
```

Or one can use a query like:

```
MonClusterLoad
| where event == 'node_state_report'
| where ClusterName startswith "tr23."
| where node_name == 'DB.57'
| extend status = iff(node_status == 'Up', 1, 0)
| project TIMESTAMP, status
| render scatterchart
```

In this case first thing to check is if there was any Platform Maintenance actions on the node or any Node Repair actions.

If the server restarted due to the node restart, and the overall failover time is small (less than 3 mins) you can share the below RCA:

> The Azure database for PostgreSQL is running as a service within Azure ecosystem, each consisting of multiple VMs and is continuously being monitored by our Azure Infrastructure and whenever unhealthy are automatically taken out of usage for repair. When this occurs, databases hosted on that VM are failed-over to their secondaries to continue processing requests. the downtime is from <start time> to <end time>. The node can be down due to various reasons viz hardware failure, OS crashes or bugs, kernel driver issues, transient race conditions etc., and as a PaaS service, it is more important for us to handle the node down promptly. In this case downtime is <x seconds or minutes> and failover completed in <x seconds or minutes>. Unfortunately, our internal telemetry couldn't detect the exact root cause of the node failure in this scenario. We cannot afford to add intensive kernel instrumentation since it can negatively impact the stability of the server. We would suggest you to add the retry logic to reduce the impaction to the application or alternatively leverage pgBouncer which has built-in retry logic and connection resiliency to handle such transient failure when the server fails over.

PLB

TSG: PLB Movement causing app restart (Web view)

```
WinFabLogs
| where ClusterName contains 'tr6' and EventType == 'Operation'
| extend Phase   = extract('Phase: ([a-z|A-Z]+)', 1, Text, typeof(string))
| extend Action  = extract('Action: ([a-z|A-Z]+)', 1, Text, typeof(string))
| extend Service = extract('Service: (.*) \r\tDecisionId: ', 1, Text, typeof(string))
| extend DecId   = extract('DecisionId: ([a-z|A-Z|0-9|-]+)', 1, Text, typeof(string))
| extend ParId   = extract('PartitionId: ([a-z|A-Z|0-9|-]+)', 1, Text, typeof(string))
| extend SrcNode = extract('SourceNode: ([a-z|A-Z|0-9|-]+)', 1, Text, typeof(string))
| extend TgtNode = extract('TargetNode: ([a-z|A-Z|0-9|-]+)', 1, Text, typeof(string))
| extend AppName = extract('fabric:/[a-z|A-Z|.]+/([a-z|A-Z|0-9]+)/', 1, Service,typeof(string))
| where Service contains 'fabric:/Worker.'
| where Phase contains 'Balancing' or Phase contains 'Constraint'
| extend EWT = ETWTimestamp
| order by PreciseTimeStamp desc nulls last
```

Customer Restart

```
MonRdmsPgSqlSandbox
| where originalEventTimestamp >= ago(1d)
| where LogicalServerName == "<your-server-name>"
| summarize min(originalEventTimestamp), max(originalEventTimestamp) by process_id, code_package_version, Logi
```

Query to find out Customer triggered restarts from Azure Portal:

```
AlrManagement
| where TIMESTAMP >= ago(1d) and elastic_server_name == "<ServerName>"
| where operation_type == 'RestartElasticServer'
| where event in ('management_operation_elastic_server')
| project originalEventTimestamp, ClusterName, operation_type, elastic_server_name, server_type, state, reques
```

Consolidated Failover RCA

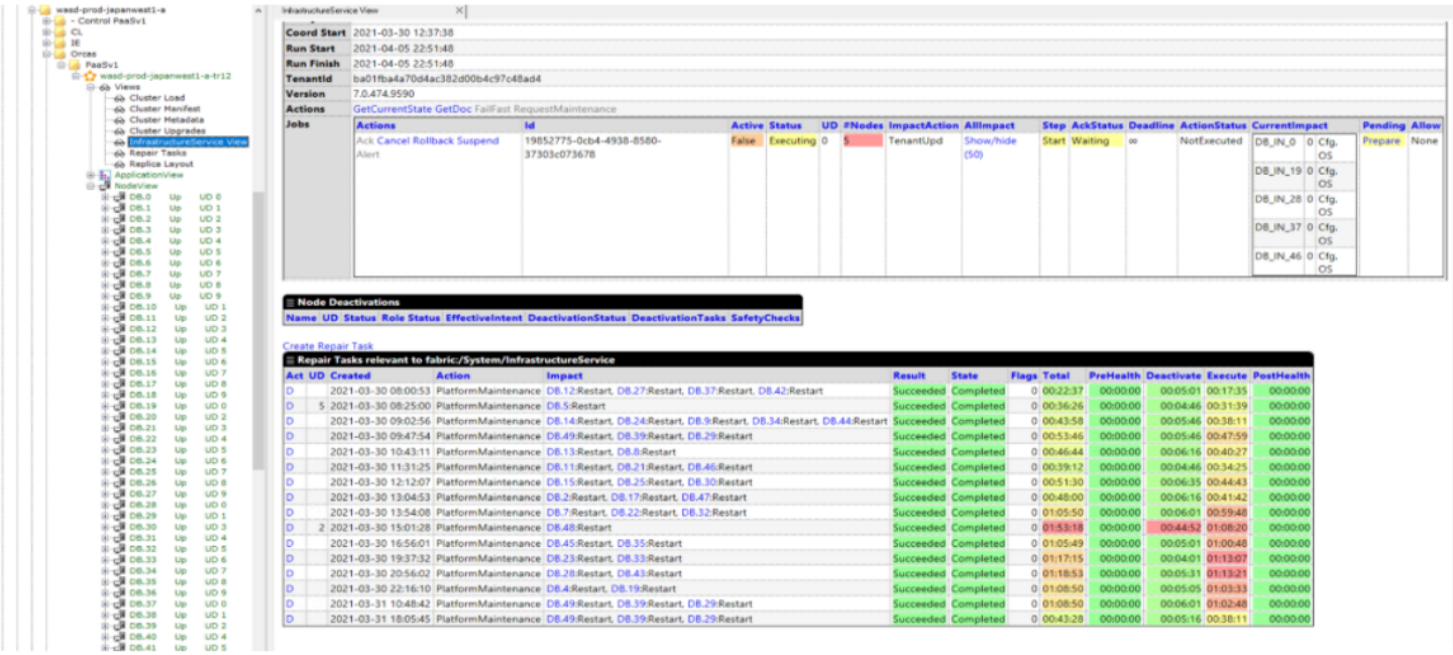
```
let TimeCheckStart = datetime(2019-03-02);
let TimeCheckEnd = datetime(2019-03-06);
let deploymentRca = 'deployment';
let plbRca = 'plb';
let userInitiatedRestartRCA = 'RestartServer';
let unknownRca = 'unknown';
let clusterName = 'tr34.eastus2-a.worker.database.windows.net';
let orcasEvents =
WinFabLogs
| where ETWTimestamp  >= TimeCheckStart and ETWTimestamp  <= TimeCheckEnd
| where ClusterName =~ clusterName
| where isnotempty(Id)
| where EventType == 'FTUpdate'and TaskName == 'FM'
| extend IdLower = tolower(Id)
| extend AppName = extract('fabric:/[a-z|A-Z|.]+/([a-z|A-Z|0-9]+)', 1, Text, typeof(string))
| where isnotempty(AppName)
| project IdLower, AppName
| join kind= leftouter ( MonAnalyticsElasticServersSnapshot
| where TIMESTAMP  >= TimeCheckStart - 40m and TIMESTAMP  <= TimeCheckEnd + 40m
| summarize by name, elastic_server_type, physical_instance_name
| extend AppName = physical_instance_name
| where isnotempty(AppName) or isnotnull(AppName)
| extend LogicalServerName = name
| extend AppTypeName = elastic_server_type
) on AppName
| project IdLower, AppName, LogicalServerName, AppTypeName
| summarize by IdLower, AppName, LogicalServerName, AppTypeName;
let RAReconfigEvents =
WinFabLogs
| where ETWTimestamp  >= TimeCheckStart and ETWTimestamp  <= TimeCheckEnd
| where isnotempty(Id)
| where ClusterName =~ clusterName
| extend IdLower = tolower(Id)
| where TaskName == 'RA' and EventType == 'ReconfigurationCompleted' and Text !contains 'EventInstanceId: '
| where Text contains '.Server.RS' and Text contains 'Result = Completed'
| extend RecoveryTime = extract('Phase2Duration: ([0-9.]+)ms', 1, Text, typeof(real)) / 1000.0
| extend currentEpoch = extract('Epoch = ([0-9:A-Za-z]+)', 1, Text)
| extend Type = extract(' Type = ([0-9A-Za-z]*)', 1, Text)
| extend Type = iff(isempty(Type), extract(' ReconfigurationType = ([0-9A-Za-z]*)', 1, Text), Type)
| extend TotalDuration = extract('TotalDuration: ([0-9.]+)ms', 1, Text, typeof(real))
| extend Phase3 = extract('Phase3Duration: ([0-9.]+)ms', 1, Text, typeof(real))
| extend Phase4 = extract('Phase4Duration: ([0-9.]+)ms', 1, Text, typeof(real))
| extend OutageStartTime = datetime_add('millisecond', toint(TotalDuration) * -1, ETWTimestamp)
| extend OutageEndTime = datetime_add('millisecond', toint(Phase3) * -1, datetime_add('millisecond', toint(Pha
| extend NewPrimary = NodeName
| extend epoch_config_number = extract('[0-9]+:([a-zA-Z0-9]+)', 1, currentEpoch)
| project OutageStartTime, OutageEndTime, currentEpoch, RecoveryTime, IdLower, Type, NewPrimary, epoch_config_
let allFmReconfigs =
WinFabLogs
| where ETWTimestamp  >= TimeCheckStart and ETWTimestamp  <= TimeCheckEnd and isnotempty(Id)
| where ClusterName =~ clusterName
| extend IdLower = tolower(Id)
| where TaskName == 'FM' and EventType == 'ReconfigurationStarted'
| extend epoch_number = extract('Reconfiguration [- a-zA-Z0-9]+/([0-9]+).', 1, Text, typeof(long))
| extend epoch_config_number = tohex(epoch_number)
| join kind= inner (
WinFabLogs
    | where ETWTimestamp  >= TimeCheckStart and ETWTimestamp  <= TimeCheckEnd and isnotempty(Id)
    | where ClusterName =~ clusterName
    | extend IdLower = tolower(Id)
    | where TaskName == 'FM' and EventType == 'ReconfigurationCompleted'
    | extend epoch_config_number = tohex(extract('Reconfiguration [- a-zA-Z0-9]+/([0-9]+).', 1, Text, typeof(l
    ) on epoch_config_number, IdLower
| extend FMReconfigStartTime = ETWTimestamp, FMReconfigEndtime = ETWTimestamp1
| extend IdLower = iff(isempty(IdLower), IdLower1, IdLower)
| where isnotempty(FMReconfigStartTime) and isnotempty(FMReconfigEndtime)
| project FMReconfigStartTime, FMReconfigEndtime, epoch_config_number, IdLower, ClusterName, AppName;
let allFailovers =
```

```
RAReconfigEvents
| join kind= fullouter allFmReconfigs on IdLower, epoch_config_number
| where Type != 'Other'
| extend OutageStartTime = iff(isempty(OutageStartTime), FMReconfigStartTime, OutageStartTime)
| extend OutageEndTime = iff(isempty(OutageEndTime), FMReconfigEndtime, OutageEndTime)
| extend IdLower = iff(isempty(IdLower), IdLower1, IdLower)
| extend ClusterName = iff(isempty(ClusterName), ClusterName1, ClusterName)
| project OutageStartTime, OutageEndTime, currentEpoch, RecoveryTime, IdLower, Type, NewPrimary, epoch_config_
| order by OutageStartTime asc;
let orcasFailovers =
allFailovers
| join kind= leftouter orcasEvents on IdLower
 | project OutageStartTime, OutageEndTime, currentEpoch, RecoveryTime, IdLower, Type, NewPrimary, epoch_config
let plbInitiatedFailovers = WinFabLogs
| where ETWTimestamp  >= TimeCheckStart and ETWTimestamp  <= TimeCheckEnd
| where isnotempty(Id)
| where ClusterName =~ clusterName
| where TaskName == 'CRM' and EventType == 'Operation'
| where Text contains 'Phase: ' and Text contains 'Action: ' and Text contains 'DecisionId: '
| extend IdLower = tolower(Id)
| extend ServiceName = extract('Service: ([\\.\\-/:a-zA-Z|0-9]+) \r', 1, Text , typeof(string)),
           DecisionId = toguid(extract('DecisionId: ([[a-z|A-Z|0-9|-]+)\r', 1, Text , typeof(string))),
           Phase = extract('Phase: ([A-Za-z]+) \r', 1, Text , typeof(string)),
           Action = extract('Action: ([A-Za-z]+) \r', 1, Text , typeof(string)),
           SourceNodeId = extract('SourceNode: ([a-zA-Z0-9]+)\r', 1, Text , typeof(string)),
           TargetNodeId = extract('TargetNode: ([a-zA-Z0-9]+)\r', 1, Text , typeof(string))
| where Action contains 'primary'
| project ETWTimestamp , ClusterName , DecisionId, Phase , Action , SourceNodeId , TargetNodeId, IdLower, Serv
| join kind = leftouter (
         WinFabLogs
    | where ETWTimestamp >= TimeCheckStart and ETWTimestamp <= TimeCheckEnd
    | where ClusterName =~ clusterName
    | where TaskName == 'PLB' and EventType == 'SchedulerAction'
    | where Text contains 'Constraint Violations: ' and ((Text contains ' NodeCapacity' and Text contains '[Me
    | extend DecisionId = toguid(extract('DecisionId: (.+)[[:space:]]Affects Service', 1, Text , typeof(string
               ConstraintType = extract('--\\[([a-zA-Z0-9_]+), [0-9]+, [0-9]+, [a-zA-Z0-9]+, [A-Z0-9\\.]+, N/
    | extend ConstraintType = iff(isempty(ConstraintType) and Text contains 'Affinity' and Text !contains '[Me
                              'Affinity', ConstraintType)
    | project ClusterName , DecisionId , ConstraintType
) on ClusterName, DecisionId
| extend PLBTimestamp = ETWTimestamp
| project PLBTimestamp, DecisionId, IdLower, Action, Phase, ConstraintType, ClusterName;
orcasFailovers
| join kind= leftouter plbInitiatedFailovers on IdLower
| extend  datediff =  datetime_diff('second', OutageStartTime, PLBTimestamp)
| extend PLBActions = iff(abs(datediff) < 120, strcat(PLBTimestamp, ':', Phase, ' ', Action, ' ', ConstraintTy
| summarize PLBActions = makeset(PLBActions) by Type, IdLower, OutageStartTime, OutageEndTime, RecoveryTime, N
| extend RCA =
    iff(Type contains 'primary' or isempty(Type),
        iff(PLBActions contains 'Upgrade', deploymentRca,
            iff(PLBActions contains 'ConstraintCheck', plbRca,
            iff(PLBActions contains 'ClientApiMovePrimary', userInitiatedRestartRCA,
            iff(PLBActions contains 'NewReplicaPlacementWithMove' or PLBActions contains 'CreationWithMove', i
            iff(PLBActions contains 'Balancing', plbRca, unknownRca))))), unknownRca)
| extend FailoverType = iff(Type =~ 'SwapPrimary' or RCA =~ deploymentRca or RCA =~ plbRca, 'Planned', 'Unplan
| extend OutageStartTime = OutageStartTime, OutageEndTime = OutageEndTime
| project OutageStartTime, OutageEndTime, RecoveryTime, NewPrimary, RCA, FailoverType, ClusterName, IdLower, T
```

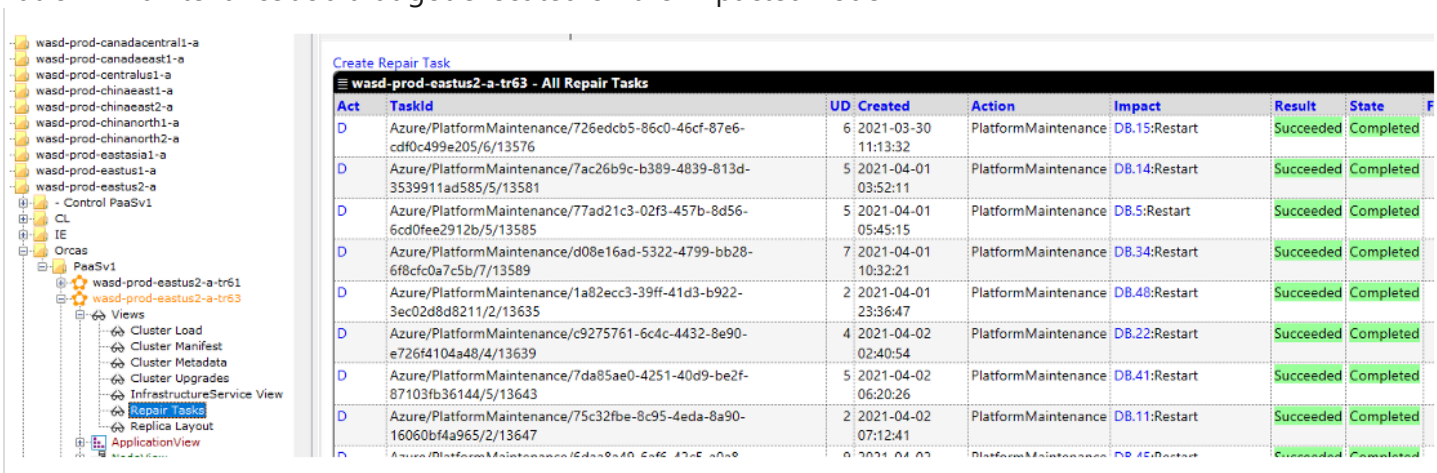
Node Restart actions

Platform Maintenance

Go to the SFE for the ring and check the "InfrastructureServices view" and see if you notice any repair tasks happened on the node.



If you see some maintainence task that happened like above, go to Repair Tasks view and see if there was a Platform Maintenance Job that got executed on the impacted node



You can also check the same in **sterling\infrawinfabstatus.xts** view in XTS by selecting the ring and going to Infra history tab.

Ring Status | Apps not ready debugging | Rollout Blockers | Drain block debugging | Pending Platform Updates | Maintenance information | Ring commands

Infra history

date_time	Job Id	State	Azure State	UD	Job Type	Notification	Impact
4/6/2021 11:20:22 PM	f42e3eda-7c96-41ca-8479-f3f5429b96df	Executing	WaitingForAcknowledgement	0	PlatformMaintenance	ImpactStart	[{"ImpactTypes":["Reboot"
3/27/2021 12:09:01 AM	5954d02c-84a1-4f97-b593-ace1a46ef91f	Executing	WaitingForAcknowledgement	0	TenantUpdate	ImpactStart	[{"ImpactTypes":["Configu
3/27/2021 12:07:58 AM	5954d02c-84a1-4f97-b593-ace1a46ef91f	Pending		-1	TenantUpdate		[]
3/27/2021 12:07:58 AM	ffed9e41-1a01-4872-a8e5-0e3d9dd92619	Executing	WaitingForAcknowledgement	0	TenantUpdate	ImpactEnd	[{"ImpactTypes":["Configu
3/14/2021 7:11:00 AM	e76eb5bd-1592-4cdb-b379-9acf2e62c5d5	Completed		-1	TenantMaintenance		[]
3/14/2021 7:09:58 AM	e76eb5bd-1592-4cdb-b379-9acf2e62c5d5	Executing	WaitingForAcknowledgement	0	TenantMaintenance	ImpactEnd	[{"ImpactTypes":["Reboot"
3/14/2021 7:05:09 AM	e76eb5bd-1592-4cdb-b379-9acf2e62c5d5	Executing	Acknowledged	0	TenantMaintenance	ImpactStart	[{"ImpactTypes":["Reboot"
3/10/2021 5:50:14 PM	5325a484-8e17-4d7e-92ca-6b0d17214da8	Failed	WaitingForAcknowledgement	0	PlatformMaintenance	ImpactStart	[{"ImpactTypes":["Reboot"
3/9/2021 7:17:44 AM	5325a484-8e17-4d7e-92ca-6b0d17214da8	Executing	WaitingForAcknowledgement	0	PlatformMaintenance	ImpactStart	[{"ImpactTypes":["Reboot"
3/8/2021 10:13:31 PM	b41924b0-728e-478f-9246-ff1843879859	Failed	WaitingForAcknowledgement	0	PlatformMaintenance	ImpactStart	[{"ImpactTypes":["Reboot"
3/6/2021 10:12:58 PM	b41924b0-728e-478f-9246-ff1843879859	Executing	WaitingForAcknowledgement	0	PlatformMaintenance	ImpactStart	[{"ImpactTypes":["Reboot"
3/6/2021 12:48:55 PM	9d4f128d-019e-4825-b510-6b349ab1ca76	Failed	WaitingForAcknowledgement	0	PlatformMaintenance	ImpactStart	[{"ImpactTypes":["Reboot"
3/4/2021 12:48:38 PM	9d4f128d-019e-4825-b510-6b349ab1ca76	Executing	WaitingForAcknowledgement	0	PlatformMaintenance	ImpactStart	[{"ImpactTypes":["Reboot"
3/3/2021 11:09:38 PM	ab0b7e29-fe4e-4a29-814f-78ce46044ab8	Failed	WaitingForAcknowledgement	0	PlatformMaintenance	ImpactStart	[{"ImpactTypes":["Reboot"
3/1/2021 11:09:27 PM	ab0b7e29-fe4e-4a29-814f-78ce46044ab8	Executing	WaitingForAcknowledgement	0	PlatformMaintenance	ImpactStart	[{"ImpactTypes":["Reboot"
3/1/2021 1:22:10 PM	a3a0d0f3-36a7-4c58-a1ce-7fce102ddbfd	Failed	WaitingForAcknowledgement	0	PlatformMaintenance	ImpactStart	[{"ImpactTypes":["Reboot"
2/27/2021 1:21:53 PM	a3a0d0f3-36a7-4c58-a1ce-7fce102ddbfd	Executing	WaitingForAcknowledgement	0	PlatformMaintenance	ImpactStart	[{"ImpactTypes":["Reboot"
2/27/2021 5:23:40 AM	588adfff-5fb3-4b52-ac89-bdab06328644	Failed	WaitingForAcknowledgement	0	PlatformMaintenance	ImpactStart	[{"ImpactTypes":["Reboot"

Active Block Modes - tr12736 | Historic block modes | Infra history | Infra Jobs | Batched Upgrade Triage | Infra Job Denied | PlatformBatching | CMS | Per-ring command | SF history | Ring

To know which team was responsible for a particular platform maintenance job, get the jobid from SFE and xts as shown above and use the below query:

```
[cluster('azurecm.kusto.windows.net').database('AzureCM')]
TMMgmtTenantManagementJobInfoEtwTable
| where JobID in ("5b57bf1f-2e6b-420f-bf77-a750f620d13d", "06eec780-fb73-4192-a223-e06168973cde")
| distinct ResponsibleTeam, JobID
| project ResponsibleTeam, JobID
| order by JobID
```

RCA draft for platform maintenance

```
PG Servers:
  Server: xxxx(region)

DESCRIPTION:
Incident timeline:
 - At 2021-0xxxx xxxx UTC, the PG server was unavailable
 - The server was available again at 2021-0xxxx UTC
 - The total restart is about xx minutes.

Root cause:
After investigation of your server instance listed above, we learnt that the unavailability was caused by an A

We follow a deployment schedule of about once a month, where we refresh database binaries and take care of oth

Mitigation & solution:
Failovers causing restarts due to deployments recover without intervention and need no mitigation.  Server ins
```

Node Repair

Go to SFE and get the Fabric Controller name for the ring

Cluster wasd-prod-centralus1-a-tr281

Cluster Properties	
Health State	Warning
Provider	SQL Config Store Provider
Connection endpoints	tr281.centralus1-a.worker.database.windows.net:9003
Credentials	ClaimsCreds(Interactive, Url: wasd-prod-centralus1-a-tr281.cloudapp.net tr281.wasd.core.windows.net)
Cluster state	Live
ClusterShortName	centralus1-a
Storage account	wasd2prodcus1atr281
Properties	ProductFamily=Orcas, Tenant
ClusterType	Production
Datacenter	DM1
MdsEndpoint	https://production.diagnostics.monitoring.core.windows.net
FabricClusterId	tr281
Fabric controller	DM1PRDAPP04

Or use MonLogin to find out the node id.

Lets assume node is DB.16, you have to use DB_IN_16 for roleInstanceName

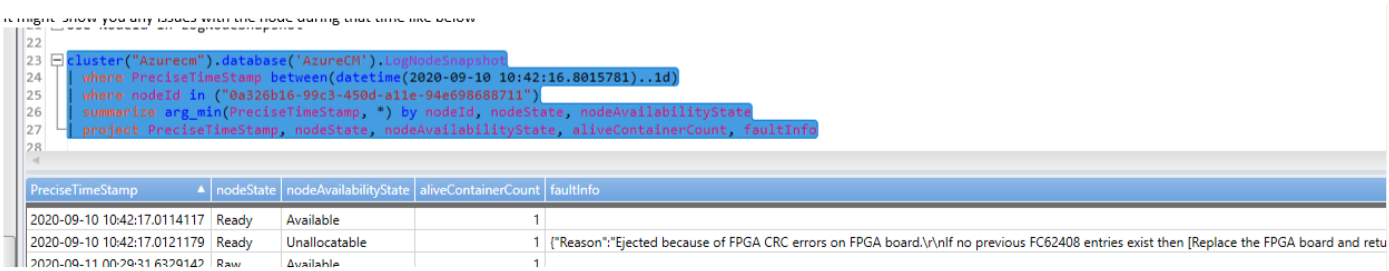
1. Use this below query to get the nodeId

```
cluster("Azurecm").database('AzureCM').LogContainerHealthSnapshot
| where PreciseTimeStamp between(datetime(2020-09-10 10:42:16.8015781)..2h)
| where Tenant =~ "SYD27PrdApp02" and roleInstanceName == "DB_IN_16"
| project PreciseTimeStamp, Tenant, nodeId, containerId, roleInstanceName
```

2. Using the nodeId guid gathered in above query, check the LogNodeSnapshot table in AzureCM cluster (https://azurecm.kusto.windows.net)

```
cluster("Azurecm").database('AzureCM').LogNodeSnapshot
| where PreciseTimeStamp between(datetime(2020-09-10 10:42:16.8015781)..1d)
| where nodeId in ("0a326b16-99c3-450d-a11e-94e698688711")
| summarize arg_min(PreciseTimeStamp, *) by nodeId, nodeState, nodeAvailabilityState
| project PreciseTimeStamp, nodeState, nodeAvailabilityState, aliveContainerCount, faultInfo
```

It might show you any issues with the node during that time like below



3. Using the nodeId guid gathered in above query, check the AnvilRepairService table in AzureCM cluster (https://azurecm.kusto.windows.net)

```
cluster("Azurecm").database('AzureCM').AnvilRepairServiceForgeEvents
| where PreciseTimeStamp between(datetime(2020-09-10 10:42:16.8015781)..1d)
| where ResourceDependencies contains "0a326b16-99c3-450d-a11e-94e698688711" and TreeActionInput != ""
| project PreciseTimeStamp, TreeNodeKey, TreeActionName, TreeActionInput
```

It might show any repair actions taken on the node like below.

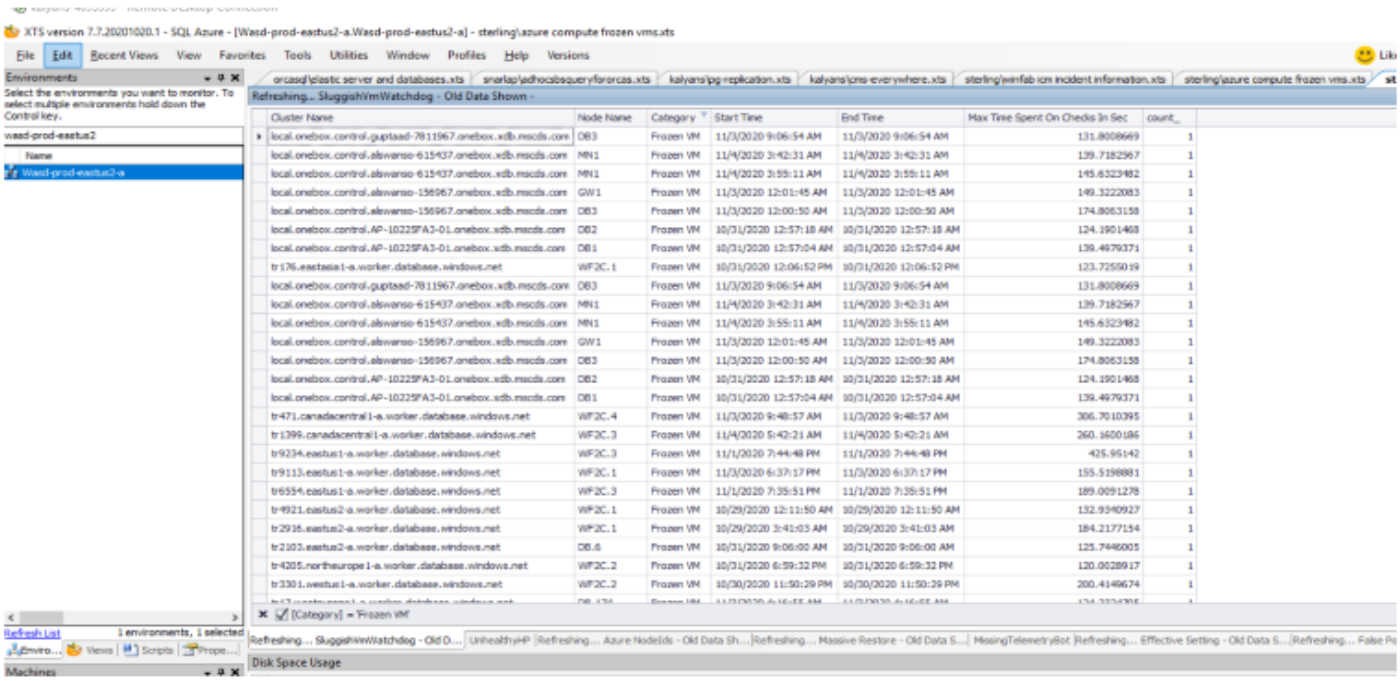
image.png

4. Use NodeId Guid to run the following query in kusto cluster https://rdos.kusto.windows.net/rdos . This can give more information about what happened on the node.

```
WindowsEventTable
| where NodeId == "ba4ef404-7997-38ac-859a-df4aa332de97"
| where PreciseTimeStamp  between (datetime(2021-01-05 00:30)..datetime(2021-01-05 02:30))
| where Level < 4
| order by PreciseTimeStamp desc
| project PreciseTimeStamp, Level, Description
```

Frozen VM Issue

This view might show some info on frozen VMs - XTS View Name: "azure compute frozen vms.xts"



How good have you found this content?