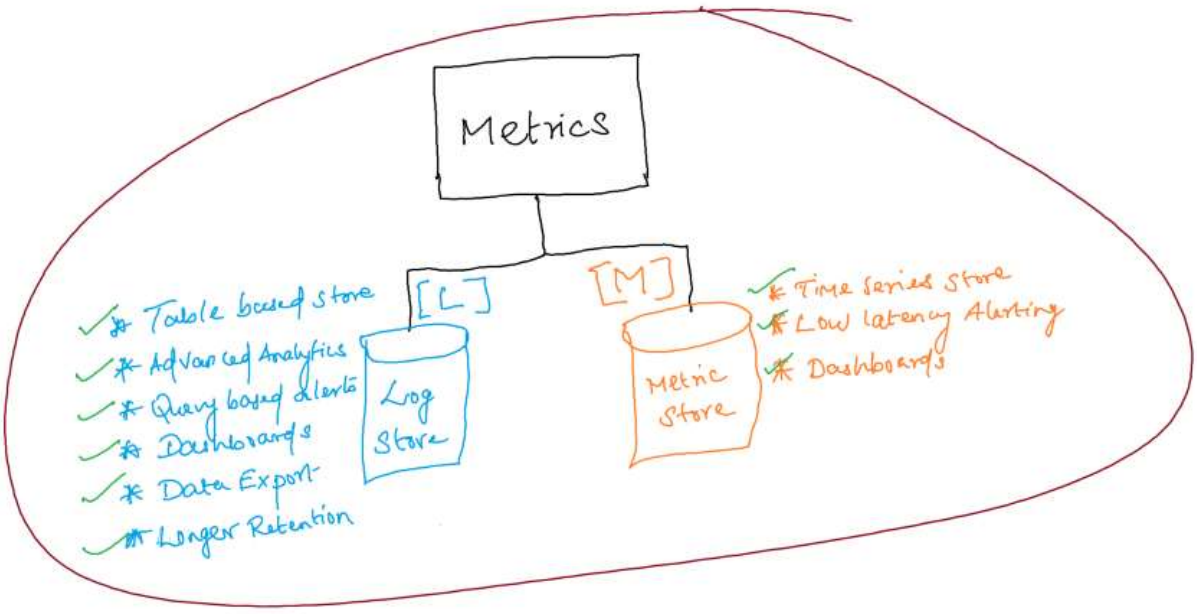
Azure monitor for containers — metrics & alerts explained

Vishwanath
Oct 21, 2020 · 12 min read

Azure monitor for containers sources good collection of metrics as part of monitoring Kubernetes clusters. This story lists out all the metrics collected in different categories by Azure monitor for containers, and also explains a little on few interesting metrics as needed. We will also touch on alerts, that can be enabled out of the box with a few clicks in the portal for any AKS cluster.

Metrics are stored in two stores by azure monitor for containers as shown below.



Azure monitor for containers — Metrics

1.Metrics stored in Azure Monitor Log analytics store — These are stored in a few 'tables' in log analytics workspace. These are billed per GB, as log analytics data

ingested. These metrics can be queried/charted/alerted (just as any log analytics data) and one can also do advanced analytics/correlation between data in different tables containing metrics, inventory & logs. One needs to write KQL/Kusto queries to setup query based alerts on these 'metrics'. I will 'tag' each of log based metric in this article as [L].

> *'**Perf**' and '**InsightsMetrics**' are the two primary tables in Log Analytics store, where [L] metrics collected by Azure monitor for containers are stored. Perf table is more like a legacy table (inspired by Windows performance counter schema), and InsightsMetrics is a newer table with more simple/modern schema to store time-series like records. At some point in the future, we will be migrating metrics from Perf table to InsightsMetrics table. Also most (almost all) of the newer metrics are stored in InsightsMetrics table.*

2. Metrics stored in Azure Monitor metrics store — These are stored as time-series data in the Azure Monitor metrics store. Azure native alerting is built on top of this platform. These metrics are not billed currently, as this is currently in 'preview'. Azure monitor for containers started to adopt this store, and you will see more metrics being routed to this store in the future. These metrics are not directly query able, and if you go to 'metrics' menu for any AKS cluster (in supported regions) in the Azure portal, there is a rich exploration/charting/alerting experience for slicing & filtering these metrics for exploration. Its super easy to enable alerting on these metrics without writing any queries. I will 'tag' each of this category of metric in this story as [M].

> *[M] metrics are available only for managed clusters (AKS, ARO, ARC etc…) and not for on-premise/unmanaged, non-Azure clusters. They are also currently not available in all Azure regions & clouds, but soon expected to be available across all regions & clouds, when it graduates from 'preview' to become generally available.*

There is no ability to corelate between [L] and [M] metrics, yet, as they are in two different backend platforms.

Metric collected by Azure monitor for containers

Azure monitor for containers monitors the below artifacts/objects in every Kubernetes cluster and collects metrics for them. These metrics are collected out of the box without requiring any configuration changes or opt-ins.

Also, metrics are collected across all namespaces for pods & containers that are monitored by Azure monitor for containers.

1. Nodes
   - CPU
   - Memory
   - GPU [Linux nodes only]
   - Disk usage (capacity) [Linux nodes only]
   - Disk IO [Linux nodes only]
   - Network [Linux nodes only]
   - and more …
2. Kubelet (running in each node)
3. Pods
4. Containers
   - CPU
   - Memory
   - GPU
   - PV Usage
   - Restarts
   - …and more…
5. Controllers
   - Kubernetes Deployments
   - HPAs (Horizontal Pod autoscalers)
6. PVs (Persistent Volumes)
   - Usage (capacity)
7. Derived metrics and more….
8. Custom/app metrics through Prometheus integration

Now, lets get into each of the above categories to see what metrics are actually collected. I will number all these metrics in increasing order across categories globally, so its easy to refer & read. For [L] metrics, I will specify the table name next to metric name inside {}. So the convention is <metricname> (metricType) [destination store] {tablename applicable only for Loganalytics store}

ex:- ***cpuUsedPercentage*** (gauge) [L] {InsightsMetrics}

> Unless otherwise noted, all metrics are collected at 60 seconds interval, and these are not configurable, yet, to make them more/less frequently sampled/collected.

Node — CPU metrics:

1. ***cpuAllocatableNanoCores*** [L] (gauge) {Perf }— Amount of cpu that is allocatable by Kubernetes to run pods, expressed in nanocores/nanocpu unit (1 nanocore = 1/1000000000th of a core/cpu; i.e 1 billionth of a core [there are *9 zeros* here if you are still counting with your mouse cursor 😀]. The entire available CPU (& memory) capacity cannot be used to run pods. There are resources reserved (depending on node type & k8s cluster configuration)for Kubelet & OS

> *Reserved resources includes -*
> ** Resources for OS & system daemons*
> ** Resources for container run-time & Kubelet*
> ** Hard eviction threshold (to avoid system/node failures like system OOM)*
>
> *AKS has detailed documentation explaining resource reservation in AKS.*

2. ***cpuCapacityNanocores*** [L] (gauge) {Perf} — Total CPU capacity of the node in nanocore/nanocpu unit

3. ***cpuUsageNanocores*** [L] (gauge) {Perf} — CPU used by node in nanocore/nanocpu unit

4. ***cpuUsageMillicores*** [M] (gauge) — CPU used by node in millicore units

5. ***cpuUsagePercentage*** [M] (gauge) — CPU used by node in percentage unit

Node — Memory metrics:

6. ***memoryAllocatableBytes*** [L] (gauge) {Perf} — Amount of memory in bytes that is allocatable by kubernetes to run pods. Like CPU, entire memory is not available for running pods due to reserved limits for OS & others (see above tip for CPU)

7. ***memoryCapacityBytes*** [L] (gauge) {Perf} — Total memory capacity of the node in bytes

8. ***memoryRssBytes*** [L] (gauge) {Perf} — Rss memory used by the node in bytes

9. ***memoryRssBytes*** [M] (gauge) — Rss memory used by the node in bytes

10. ***memoryRssPercentage*** [M] (gauge) {Perf} — Rss memory used by the node in percentage unit

> *memoryRss* metrics are collected only for Linux nodes*

11. ***memoryWorkingSetBytes*** [L] (gauge) {Perf} — Working set memory used by the node in bytes

12. ***memoryWorkingSetBytes*** [M] (gauge) — Working set memory used by the node in bytes

13. ***memoryWorkingSetPercentage*** [M] (gauge) — Working set memory used by the node in percentage units

Node — Other metrics:

14. ***restartTimeEpoch*** [L] (gauge) {Perf} — Last time node restarted in epoch seconds

15. ***nodesCount*** [M] (gauge) — count of nodes by last know status

Node — Disk usage metrics:

16. ***free*** [L] (gauge) {InsightsMetrics} — Free disk space in bytes

17. ***used*** [L] (gauge) {InsightsMetrics} — Used disk space in bytes

18. ***used_percent [L]*** (gauge) {InsightsMetrics} — Used disk space in percentage

19. ***diskUsedPercentage*** [M] (gauge) — Used disk space per disk in percentage

> *For disk usage metrics, Azure monitor for containers does not collect metrics for the following file system types — tmpfs, devtmpfs, devfs, overlay, aufs, squashfs to limit noise.*

Node — Disk IO metrics:

20. ***reads*** [L] (counter) {InsightsMetrics} — Number of reads (incremented when I/O request completes)

21. ***read_bytes*** [L] (counter) {InsightsMetrics} — Number of bytes read from the block device

22. ***read_time*** [L] (counter) {InsightsMetrics} — Number of milliseconds that read requests have waited on the block device. If there are multiple I/O requests waiting, these values will increase at a rate greater than 1000/second; for example, if 60 read

requests wait for an average of 30 ms, the read_time field will increase by 60*30 = 1800

23. ***writes*** [L] (counter) {InsightsMetrics} — Number of writes (incremented when I/O request completes)

24. ***write_bytes*** [L] (counter) {InsightsMetrics} — Number of bytes written to the block device

25. ***write_time*** [L] (counter) {InsightsMetrics} — Number of milliseconds that write requests have waited on the block device. If there are multiple I/O requests waiting, these values will increase at a rate greater than 1000/second; for example, if 60 write requests wait for an average of 30 ms, the write_time field will increase by 60*30 = 1800

26. ***io_time*** [L] (counter) {InsightsMetrics} — Number of milliseconds during which the device has had I/O requests queued

27. ***iops_in_progress*** [L] (gauge) {InsightsMetrics} — Number of I/O requests that have been issued to device driver but have not yet completed. It does not include I/O requests that are in the queue but not yet issued to the device driver

> *For Disk IO metrics, Azure monitor for containers filter for devices having names with the regex pattern "sd[a-z][0-9]" and include only those devices to limit noise.*
>
> *Also, for DiskIO, in Linux, the above metrics correspond to the values in /proc/diskstats and /sys/block/<dev>/stat.*

Node — GPU metrics:

28. ***nodeGpuAllocatable*** [L] (gauge) {InsightsMetrics} — Number of allocatable GPUs in the node at any point in time

29. ***nodeGPUCapacity*** [L] (gauge) {InsightsMetrics} — Total number of GPUs in the node

Node — Network metrics:

30. ***bytes_recv*** [L] (counter) {InsightsMetrics} — Total number of bytes received by the interface

31. ***bytes_sent*** [L] (counter) {InsightsMetrics} — Total number of bytes sent by the interface

32. ***err_in*** [L] (counter) {InsightsMetrics} — Total number of receive errors detected by the interface

33. ***err_out*** [L] (counter) {InsightsMetrics} — Total number of transmit errors detected by the interface

Container — CPU metrics:

34. ***cpuRequestNanoCores*** [L] (gauge) {Perf} — container's cpu request in nanocore/nanocpu unit

35. ***cpuLimitNanoCores*** [L] (gauge){Perf} — container's cpu limit in nanocore/nanocpu unit

> ** Specifying resource requests & limits is always a best practice.*
> ** If container cpu resource requests are not specified, cpuRequestNanoCores metric will not be collected*
> ** If container resource limits are not specified, node's capacity will be rolled-up as container's limit*

36. ***cpuUsageNanoCores*** [L] (gauge) {Perf} — container's CPU usage in nanocore/nanocpu unit

37. cpuExceededPercentage [M] (gauge) — Container's CPU usage exceeded threshold % [default threshold is 95% and configurable]

> *cpuExceededPercentage metric is collected only when threshold is exceeded*

Container — Memory metrics

38. ***memoryRequestBytes*** [L] (gauge) {Perf} — container's memory request in bytes

39. ***memoryLimitBytes*** [L] (gauge) {Perf} — container's memory limit in bytes

> ** Specifying resource requests & limits is always a best practice.*
> ** If container memory resource requests are not specified, memoryRequestBytes metric will not be collected*
> ** If container resource limits are not specified, node's capacity will be rolled-up as container's limit*

40. ***memoryRssBytes*** [L] (gauge) {Perf} — container's rss memory usage in bytes

41. ***memoryRssExceededPercentage*** [M] (gauge) — container's rss memory usage exceeded configured threshold % [default threshold is 95% and configurable]

> *memoryRssExceededPercentage metric is collected only when threshold is exceeded*
> *memoryRss* metrics are collected only for containers running in Linux nodes*

42. ***memoryWorkingSetBytes*** [L] (gauge) {Perf} — container's working set memory usage in bytes

43. ***memoryWorkingSetExceededPercentage*** [M] (gauge) — container's workingset memory usage exceeded configured threshold % [default threshold is 95% and configurable]

Container — Other metrics:

44. ***restartTimeEpoch*** [L] (gauge) {Perf} — Last time the container restarted in epoch seconds

Container — GPU metrics

45. ***containerGpuRequests*** [L] (gauge) {InsightsMetrics} — number of GPUs requested by the container

46. ***containerGpuLimits*** [L] (gauge) {InsightsMetrics} — container's GPU limit

47. ***containerGpuDutyCycle*** [L] (gauge) {InsightsMetrics} — Percentage of time over the past sample period during which GPU was busy/actively processing for a container. Duty cycle is a number between 1 and 100

48. ***containerGpumemoryTotalBytes*** [L] (gauge) {InsightsMetrics} — total GPU memory available for the container

49. ***containerGpumemoryUsedBytes*** [L] (gauge) {InsightsMetrics} — GPU memory used by the container

Kubelet metrics

50. ***kubelet_docker_operations*** [L] (counter) {InsightsMetrics} — Cumulative number of Docker operations by operation type

51. ***kubelet_docker_operations_errors*** [L] (counter) {InsightsMetrics} — Cumulative number of Docker operation errors by operation type

> *kubelet_docker_operations_errors & kubelet_docker_operations_errors are deprecated in later versions of k8s. They will be replaced by kubelet_docker_operations_total & kubelet_docker_operations_errors_total respectively in the future.*

52. ***kubelet_running_pod_count*** [L] (gauge) {InsightsMetrics} — Number of pods currently running

53. ***volume_manager_total_volumes*** [L] (gauge) {InsightsMetrics} — Number of volumes in Volume Manager

54. ***kubelet_node_config_error*** [L] (gauge) {InsightsMetrics} — This metric is true (1) if the node is experiencing a configuration-related error, false (0) otherwise

55. ***process_resident_memory_bytes*** [L] (gauge) {InsightsMetrics} — Kubelet's resident memory size in bytes

56. ***process_cpu_seconds_total*** [L] (counter) {InsightsMetrics} — Kubelet's total user and system CPU time spent in seconds

Pod metrics — PV (Persistent Volumes):

57. ***pvUsedBytes*** [L] (gauge) {InsightsMetrics} — Used space in bytes for a specific PV consumed by a specific Pod

> *pvUsedBytes metric has pvCapacityBytes as a dimension ('tags' column in InsightsMetrics table). Capacity is 'folded' into usage metric to save data ingestion cost and also for better/easier queryability.*

58. ***pvUsageExceededPercentage*** [M] (gauge) — Pods for which PV usage exceeded by configured threshold (60% default) by podname & namespace

More Pod metrics

59. ***completedJobsCount*** [M] (gauge) — Count of completed jobs (that are yet to be cleaned up) in the past 6 hours (default) by namespace & controller

60. ***podCount*** [M] (gauge) — Count of pods by namespace, controller & phase

61. ***podReadyPercentage*** [M] (gauge) — Percentage of pods in 'ready' state by namespace & controller

62. ***oomKilledContainerCount*** [M] (gauge) — Count of OOM killed containers by namespace & controller

63. ***restartingContainerCount*** [M] (gauge) — Count of container restarts by controller & namespace

> *You can read about how to configure thresholds for all of the [M] metrics with configurable thresholds here*

Kubernetes Deployment metrics:

64. ***kube_deployment_status_replicas_ready*** [L] (gauge) {InsightsMetrics} — Total number of ready pods targeted by deployment (status.readyReplicas)

> *kube_deployment_status_replicas_ready metric also has status_replicas_available, spec_replicas and status_replicas_updated values as dimensions (instead of individual metrics) to save ingestion cost and also for better/easier queryability. This metric is inspired by kube-state metrics project.*

HPA metrics:

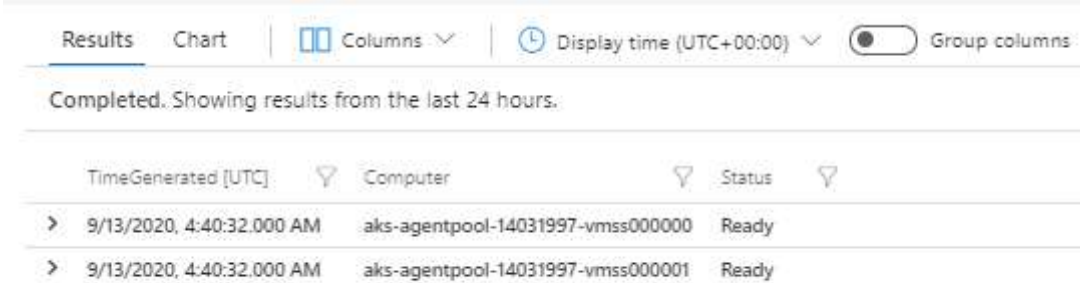
65. ***kube_hpa_status_current_replicas*** [L] (gauge) {InsightsMetrics} — Current number of replicas of pods managed by this autoscaler (status.currentReplicas)

> *kube_hpa_status_current_replicas metric also has spec_max_replicas, spec_min_replicas and status_desired_replicas values as dimensions (instead of individual metrics) to save ingestion cost and also for better/easier queryability. This metric is inspired by kube-state metrics project.*

Derived metrics:

With [L] metrics stored in Log analytics store, it is very easy to write queries and arrive at 'derived' metrics.

> *For ex;- To get the last known nodes by node status,below is the query and its result.*
>
> *KubeNodeInventory | summarize arg_max(TimeGenerated, Status) by Computer*



Last known nodes by node status

> **Tip:** *It will be very useful to look at data & schema for few of the key tables in which data is stored by Azure Monitor for Containers. These key tables include but not limited to -*
> ** KubePodInventory*
> ** KubeNodeInventory*
> ** KubeEvents*
> ** ContainerInventory*
> ** KubeServices*
> ** ContainerLog*
> ** Perf*
> ** InsightsMetrics*
>
> *You can also read about their schema which is documented here.*

Custom/App metrics:

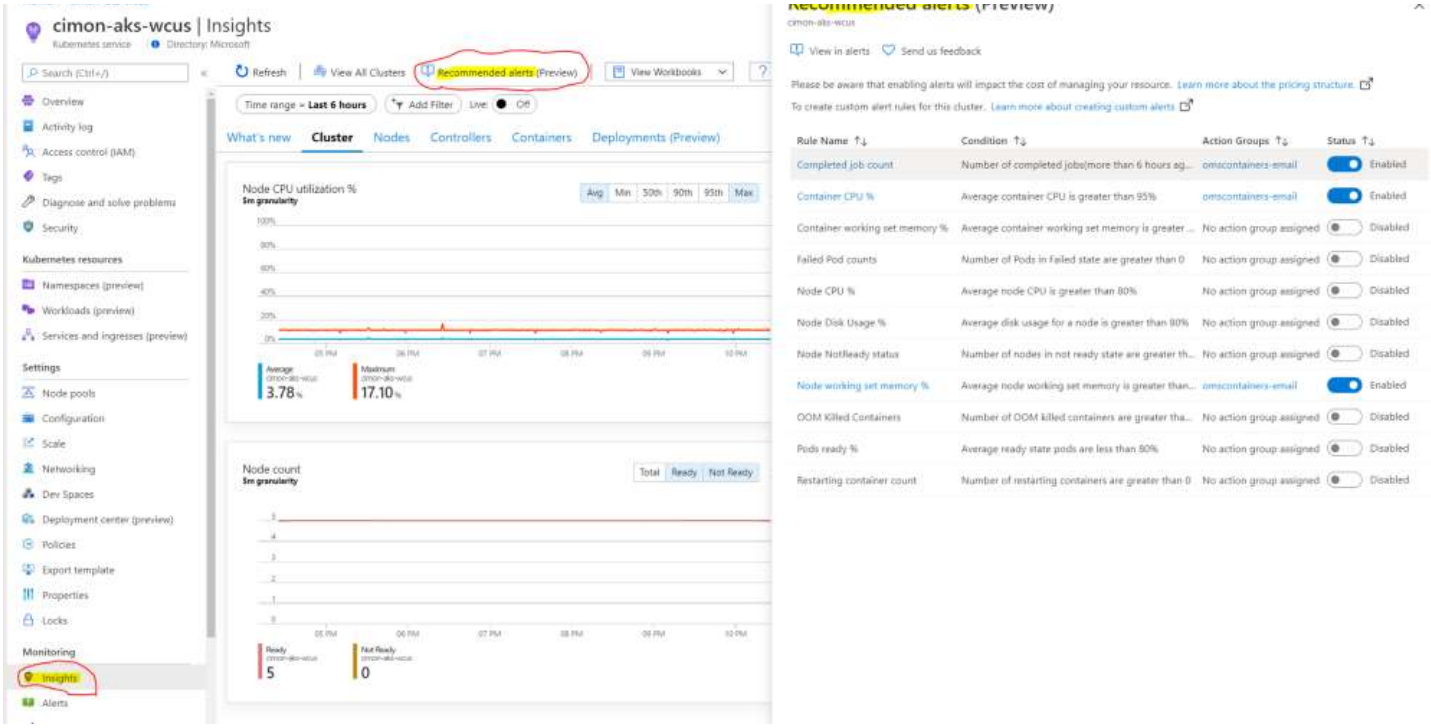
As you may know, you can also configure Azure monitor for containers to collect/scrape Prometheus metrics from pods & services running inside/outside the cluster through Prometheus integration. Currently metrics from Prometheus integration gets stored in Log Analytics store.

Alerting in Azure Monitor for Containers:

Log based alerts: One can write KQL queries on any of the [L] metrics that we saw in this story and set up alerts on them. Log based alerts are very flexible (you can join any table with any other table(s) and derive a metrics that you can alert on), but the minimum frequency to run these queries is at least 5 minutes.

Metric based alerts: From the 'metrics' menu in an AKS cluster, you can see all the [M] metrics that we saw in this story, all neatly organized by categories/namespaces. These metrics nicely fit into Azure alerting platform to provide first class/low latency alerts on them. You can setup alerts on these metrics with filtering & grouping on their dimensions. These alerts can be granular up to a minute.

Recommended alerts: You can quickly enable recommended alerts for an AKS cluster in a few clicks in the portal. These alerts are all based on [M] metrics and gets configured with default thresholds, that can be configured as needed.



Azure monitor for containers — recommended alerts feature

> *For every AKS cluster, there are alerts that are recommended by Azure monitor for containers product team in Microsoft, and you can enable these alerts (with their default thresholds) in a few clicks from the Azure Portal Ux or through Azure ARM template. More alert rules are added to recommended category every few months, and expect to see more and more here. You can read more about recommended alerts feature here.*

To summarize — If you would like to see all the above metrics in detail along with their dimensions in a tabular form, you can refer to the below gist.

Search this file…

	MetricCategory	MetricName	MetricDimensions
2	Node-CPU	cpuAllocatableNanoCores	Objectname='K8SNode', Instancename=<nodenar
3	Node-CPU	cpuCapacityNanocores	Objectname='K8SNode', Instancename=<nodenar
4	Node-CPU	cpuUsageNanocores	Objectname='K8SNode', Instancename=<nodenar
5	Node-Memory	memoryAllocatableBytes	Objectname='K8SNode', Instancename=<nodenar
6	Node-Memory	memoryCapacityBytes	Objectname='K8SNode', Instancename=<nodenar
7	Node-Memory	memoryRssBytes	Objectname='K8SNode', Instancename=<nodenar
8	Node-Memory	memoryWorkingSetBytes	Objectname='K8SNode', Instancename=<nodenar
9	Node-Other	restartTimeEpoch	Objectname='K8SNode', Instancename=<nodenar
10	Node-DiskUsage	free	device,hostName,path,clusterId,clusterName
11	Node-DiskUsage	used	device,hostName,path,clusterId,clusterName
12	Node-DiskUsage	used_percent	device,hostName,path,clusterId,clusterName
13	Node-DiskIO	reads	hostName,name,clusterId,clusterName
14	Node-DiskIO	read_bytes	hostName,name,clusterId,clusterName
15	Node-DiskIO	read_time	hostName,name,clusterId,clusterName

Node-DiskIO	writes	hostName,name,clusterId,clusterName
Node-DiskIO	write_bytes	hostName,name,clusterId,clusterName
Node-DiskIO	write_time	hostName,name,clusterId,clusterName
Node-DiskIO	io_time	hostName,name,clusterId,clusterName
Node-DiskIO	iops_in_progress	hostName,name,clusterId,clusterName
Node-GPU	nodeGpuAllocatable	gpuVendor,Computer,clusterId,clusterName
Node-GPU	nodeGPUCapacity	gpuVendor,Computer,clusterId,clusterName
Node-Network	bytes_recv	hostName,interface,clusterId,clusterName
Node-Network	bytes_sent	hostName,interface,clusterId,clusterName
Node-Network	err_in	hostName,interface,clusterId,clusterName
Node-Network	err_out	hostName,interface,clusterId,clusterName
Container-CPU	cpuRequestNanoCores	Objectname='K8SContainer', Instancename=podU
Container-CPU	cpuLimitNanoCores	Objectname='K8SContainer', Instancename=podU
Container-CPU	cpuUsageNanoCores	Objectname='K8SContainer', Instancename=podU
Container-Memory	memoryRequestBytes	Objectname='K8SContainer', Instancename=podU
Container-Memory	memoryLimitBytes	Objectname='K8SContainer', Instancename=podU
Container-Memory	memoryRssBytes	Objectname='K8SContainer', Instancename=podU
Container-Memory	memoryWorkingSetBytes	Objectname='K8SContainer', Instancename=podU
Container-Other	restartTimeEpoch	Objectname='K8SContainer', Instancename=podU
Container-GPU	containerGpuRequests	containerName=podUID/containerName,clusterId,
Container-GPU	containerGpuLimits	containerName=podUID/containerName,clusterId,
Container-GPU	containerGpuDutyCycle	containerName=podUID/containerName,gpuId,gp
Container-GPU	containerGpumemoryTotalBytes	containerName=podUID/containerName,gpuId,gp
Container-GPU	containerGpumemoryUsedBytes	containerName=podUID/containerName,gpuId,gp
Pod-PV	pvUsedBytes	podUID,podName,podNamespace,pvName,pvcNa
Controller-Deployments	kube_deployment_status_replicas_ready	creationTime,deployment,deploymentStrategy,k8s
Controller-HPA	kube_hpa_status_current_replicas	creationTime,hpa,k8sNamespace,lastScaleTime,spe
Kubelet	kubelet_docker_operations	hostName,operation_type,scrapeUrl,clusterId,cluste
Kubelet	kubelet_docker_operations_errors	hostName,operation_type,scrapeUrl,clusterId,cluste
Kubelet	kubelet_running_pod_count	hostName,scrapeUrl,clusterId,clusterName
Kubelet	volume_manager_total_volumes	hostname,plugin_name,scrapeUrl,state,clusterId,clu
Kubelet	kubelet_node_config_error	hostName,scrapeUrl,clusterId,clusterName
Kubelet	process_resident_memory_bytes	hostName,scrapeUrl,clusterId,clusterName
Kubelet	process_cpu_seconds_total	hostName,scrapeUrl,clusterId,clusterName

AzureMonitorContainers-Metrics-Loganalytics.csv hosted with ♡ by GitHub view raw

Search this file…

MetricCategory	MetricName	MetricDimensions
Node-CPU	cpuUsageMillicores	host
Node-CPU	cpuUsagePercentage	host
Node-Memory	memoryRssBytes	host
Node-Memory	memoryRssPercentage	host
Node-Memory	memoryWorkingSetBytes	host
Node-Memory	memoryWorkingSetPercentage	host
Node-Other	nodesCount	Status

	Node-Other	nodesCount	Status
9	Node-DiskUsage	diskUsedPercentage	device,host
10	Container-CPU	cpuExceededPercentage	containerName,controllerName,Kubernetes namespace
11	Container-Memory	memoryRssExceededPercentage	containerName,controllerName,Kubernetes namespace
12	Container-Memory	memoryWorkingSetExceededPercentage	containerName,controllerName,Kubernetes namespace
13	Pod	podCount	controllerName, Kubernetes namespace,node,phase
14	Pod	podReadyPercentage	controllerName, Kubernetes namespace
15	Pod	oomKilledContainerCount	controllerName, Kubernetes namespace
16	Pod	restartingContainerCount	controllerName, Kubernetes namespace
17	Pod	completedJobsCount	controllerName, Kubernetes namespace

That's all about metrics & alerting for now. We will try to keep this story up-to-date as we add more categories & metrics. Hope this was useful. Thanks for your read!

> **Please leave feedback(s) you might have as comments and you can also contact us through email (askcoin@microsoft.com) with any questions or suggestions/feedbacks you might have, including any request(s) to add new metrics and features requests/ideas**

Helpful Links :

* Azure Monitor for Containers
* Azure Monitor for Containers — FAQ
* Azure Monitor for containers — Recommended alerts
* Azure Monitor for Containers — Log analytics tables & their schema
* Azure Monitor for Containers — Agent data collection settings
* Azure Monitor for Containers — Agent configmap
* Azure Monitor for Containers — Prometheus Integration
* Azure Log analytics — managing usage & costs
* Azure Log Analytics — Query language reference
* Azure Monitor
* Azure Kubernetes Service (AKS)

Azure Azure Monitor Azure Kubernetes Service Kubernetes Monitoring