learnk8s RESEARCH

Research on the trade offs when choosing an instance type for a Kubernetes cluster

Find more research at: https://learnk8s.io/research

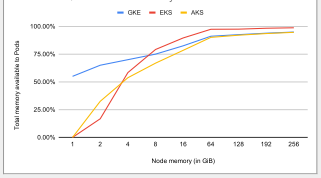
How to contribute: Leave a comment or drop us a line at research@learnk8s.io

License: Apache 2.0

Last updated: February 14, 2022

Check out the Kubernetes instance calculator

Quota of Node memory available to Pods



What is it?

Not all memory and CPU in a Node can be used to run Pods. The resources are partitioned in 4:

1. Memory and CPU reserved to the operating system and system daemons such as SSH
2. Memory and CPU reserved to the Kubelet and Kubernetes agents such as the CRI
3. Memory reserved for the hard eviction threshold
4. Memory and CPU available to Pods

The graph show the total memory available for running Pods after subtracting the reserved memory

Example

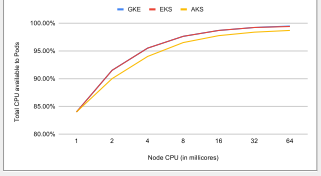
If you have a Kubernetes cluster in GKE with a single Node of 2GB of memory, only 65% of the available memory is used to run Pods. The reamining memory is necessary to run the OS, Kubelet, CRI, CNI, etc.

Notes

GKE and AKS reach 90% level of utilisation with instances over 64GB. EKS is 90% efficient starting with 8GB.

Memory (in GiB)	GKE	EKS	AKS
1	55.00%	0.00%	0.00%
2	65.00%	16.75%	32.50%
4	70.00%	58.38%	53.75%
8	75.00%	79.19%	66.88%
16	82.50%	89.59%	78.44%
64	91.13%	97.40%	90.11%
128	92.56%	97.50%	92.05%
192	93.88%	98.33%	93.54%
256	94.91%	98.75%	94.65%

Quota of Node CPU available to Pods



What is it?

Not all memory and CPU in a Node can be used to run Pods. The resources are partitioned in 4:

1. Memory and CPU reserved to the operating system and system daemons such as SSH
2. Memory and CPU reserved to the Kubelet and Kubernetes agents such as the CRI
3. Memory reserved for the hard eviction threshold
4. Memory and CPU available to Pods

The graph show the total memory available for running Pods after subtracting the reserved memory

Example

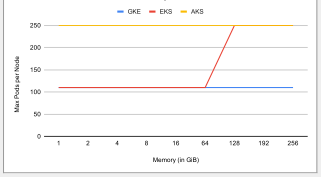
If you have a Kubernetes cluster in AKS with a single Node and 2 vCPU 90% of the available CPUs are used to run Pods. The reamining memory is necessary to run the OS, Kubelet, CRI, CNI, etc.

Notes

As long as you use node with at least 2 vCPU you should be fine.

CPU (in millicores)	GKE	EKS	AKS
1	84.00%	84.00%	84.00%
2	91.50%	91.50%	90.00%
4	95.50%	95.50%	94.00%
8	97.63%	97.63%	96.50%
16	98.69%	98.69%	97.75%
32	99.22%	99.22%	98.38%
64	99.48%	99.40%	98.69%

Max Pods per Node



What is it?

There's a upper limit on the number of Pods that you can run on each Node.

Each cloud provider has a different limit.

Most of the time the limit is independent of the Node size (e.g. GKE, AKS).

There are cases where the number of Pods depends on the Node size (notable: EKS).

Notes

The metrics is relevant to measure your blast radius.

Assuming that a Node is lost how many Pods are affected?

Memory (in GiB)	GKE	EKS	AKS
1	110	110	250
2	110	110	250
4	110	110	250
8	110	110	250
16	110	110	250
64	110	110	250
128	110	250	250
192	110	250	250
256	110	250	250

Max memory per Pod at full Node capacity

What is it?

Nodes have an upper limit on the number of Pods that they can run.

Assuming that you run the max number of Pods for that node, how much memory is available to each Pod?

This metric divides the available Node memory by the max number of Pods for that instance type.

Example

If you have Kubernetes cluster in GKE with a single Node of 128GB of memory, you can run up to 110 Pods and each of them can use 1.08GB of memory.

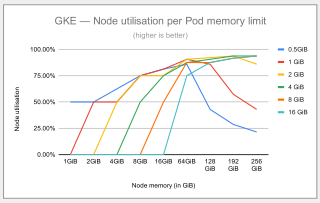
Notes

It's not possible to run small workloads (less than 1GB of memory) efficiently on GKE when the node size is greater than 128GB of memory.

EKS has a peak at 192GB of memory. That's where there are the most Pod with the larger memory available to them (234 Pods with 810MiB of memory each).

Memory (in GiB)	GKE	EKS	AKS
1	0.01	0.00	0.00
2	0.01	0.00	0.00
4	0.03	0.02	0.01
8	0.05	0.06	0.02
16	0.12	0.13	0.05
64	0.53	0.57	0.23
128	1.08	0.50	0.47
192	1.64	0.76	0.72
256	2.21	1.01	0.97

GKE — Node utilisation per Pod memory limit



What is it?

If all my Pods are using 1GB of memory, what instance type I should use to maximise the memory available?

The charts presents 5 scenarios: what if all the Pods in the Node have limits of 1, 2, 4, 8 or 16 GiB.

The chart shows how utilised is the node.

Example

When all Pods in your cluster are 1GB, the best node that can allocate the most number of Pods is a Node with 64GB of memory.

Values before the peak means that the node is underutilised (there's still space, but not enough to run a Pod).

Values after the peak means that you reached the limit of Pods on that and you can't schedule more Pods on that node.

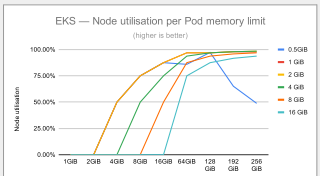
Notes

It's clear that the best Node for Pods that average 1GB of memory is a 64GB Node.

If the Pod memory limit increases in average to 2GB, a 192GB memory instance is the more efficient.

Pod memory limit	1GiB	2GiB	4GiB	8GiB	16GiB	64GiB	128 GiB	192 GiB	256 GiB
0.5GiB	50.00%	50.00%	62.50%	75.00%	81.25%	85.94%	42.97%	28.65%	21.48%
1 GiB	0.00%	50.00%	50.00%	75.00%	81.25%	90.63%	85.94%	57.29%	42.97%
2 GiB	0.00%	0.00%	50.00%	75.00%	75.00%	90.63%	92.19%	93.75%	85.94%
4 GiB	0.00%	0.00%	0.00%	50.00%	75.00%	87.50%	90.63%	93.75%	93.75%
8 GiB	0.00%	0.00%	0.00%	0.00%	50.00%	87.50%	87.50%	91.67%	93.75%
16 GiB	0.00%	0.00%	0.00%	0.00%	0.00%	75.00%	87.50%	91.67%	93.75%

EKS — Node utilisation per Pod memory limit



What is it?

If all my Pods are using 1GB of memory, what instance type I should use to maximise the memory available?

The charts presents 5 scenarios: what if all the Pods in the Node have limits of 1, 2, 4, 8 or 16 GiB.

The chart shows how utilised is the node.

Example

When all Pods in your cluster are 1GB, the best node that can allocate the most number of Pods is a Node with 192GB of memory.

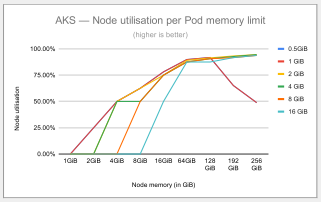
Values before the peak means that the node is underutilised (there's still space, but not enough to run a Pod).

Notes

Pay attention to local inefficiencies due to the limits on how many Pods can be deployed on a Node.

Pod memory limit	1GiB	2GiB	4GiB	8GiB	16GiB	64GiB	128 GiB	192 GiB	256 GiB
0.5GiB	0.00%	0.00%	50.00%	75.00%	87.50%	85.94%	97.27%	65.10%	48.83%
1 GiB	0.00%	0.00%	50.00%	75.00%	87.50%	96.88%	96.88%	97.92%	97.66%
2 GiB	0.00%	0.00%	50.00%	75.00%	87.50%	96.88%	96.88%	97.92%	98.44%
4 GiB	0.00%	0.00%	0.00%	50.00%	75.00%	93.75%	96.88%	97.92%	98.44%
8 GiB	0.00%	0.00%	0.00%	0.00%	50.00%	87.50%	93.75%	95.83%	96.88%
16 GiB	0.00%	0.00%	0.00%	0.00%	0.00%	75.00%	87.50%	91.67%	93.75%

AKS — Node utilisation per Pod memory limit



What is it?

If all my Pods are using 1GB of memory, what instance type I should use to maximise the memory available?

The charts presents 5 scenarios: what if all the Pods in the Node have limits of 1, 2, 4, 8 or 16 GiB.

The chart shows how utilised is the node.

Example

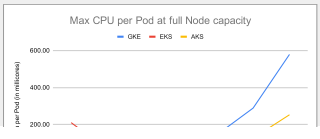
When all Pods in your cluster are 1GB, the best node that can allocate the most number of Pods is a Node with 128GB of memory.

Values before the peak means that the node is underutilised (there's still space, but not enough to run a Pod).

Values after the peak means that you reached the limit of Pods on that and you can't schedule more

Pod memory limit	1GiB	2GiB	4GiB	8GiB	16GiB	64GiB	128 GiB	192 GiB	256 GiB
0.5GiB	0.00%	25.00%	50.00%	62.50%	78.13%	89.84%	91.80%	65.10%	48.83%
1 GiB	0.00%	25.00%	50.00%	62.50%	78.13%	89.84%	91.80%	65.10%	48.83%
2 GiB	0.00%	0.00%	50.00%	62.50%	75.00%	89.06%	91.41%	93.23%	94.53%
4 GiB	0.00%	0.00%	50.00%	50.00%	75.00%	87.50%	90.63%	92.71%	94.53%
8 GiB	0.00%	0.00%	0.00%	50.00%	75.00%	87.50%	90.63%	91.67%	93.75%
16 GiB	0.00%	0.00%	0.00%	0.00%	50.00%	87.50%	87.50%	91.67%	93.75%

Max CPU per Pod at full Node capacity



What is it?

Nodes have an upper limit on the number of Pods that they can run.

Assuming that you run the max number of Pods for that node, how much CPU is available to each Pod?

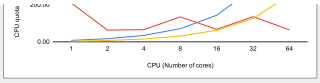
This metric divides the available Node cores by the max number of Pods for that instance type.

Example

GKE can have at most 110 pods per node.

If the Node has 16 cores, you can split the available millicores evenly to all Pods. Each Pod

CPU cores	GKE	EKS	AKS
1	7.64	210.00	3.36
2	16.64	63.10	7.20
4	34.73	65.86	15.04
8	71.00	134.66	30.88
16	143.55	67.48	62.56
32	288.64	135.68	125.92
64	578.82	64.74	252.64



Example	receives a quota of 143.55 millicores.
Notes	Larger nodes in GKE and AKS lead to Pod with more CPU available (due to the cap on the number of Pods)

	Millicores reserved for the kubelet and kubernetes a									
CPU cores	GKE									
	Kubelet	OS	Total reserved	Total reserved %	Available	Available %				
1	60	100	160	16.00%	840	84.00%			Max pods per no	110
2	70	100	170	8.50%	1830	91.50%				
4	80	100	180	4.50%	3820	95.50%				
8	90	100	190	2.38%	7810	97.63%				
16	110	100	210	1.31%	15790	98.69%				
32	150	100	250	0.78%	31750	99.22%				
64	230	100	330	0.52%	63670	99.48%				

Memory									
GKE	Kubelet	OS	Hard eviction	Total Reserved	Total Reserved %	Available for worl	Available for worl	max density (GB)	
1	0.25	0.1	0.1	0.45	45.00%	0.55	55.00%	0.005	e2-micro
2	0.5	0.1	0.1	0.7	35.00%	1.3	65.00%	0.01181818182	e2-small
4	1	0.1	0.1	1.2	30.00%	2.8	70.00%	0.02545454545	e2-medium
8	1.8	0.1	0.1	2	25.00%	6	75.00%	0.05454545455	e2-standard-2
16	2.6	0.1	0.1	2.8	17.50%	13.2	82.50%	0.12	e2-standard-4
64	5.48	0.1	0.1	5.68	8.88%	58.32	91.13%	0.5301818182	e2-standard-16
128	9.32	0.1	0.1	9.52	7.44%	118.48	92.56%	1.077090909	e2-standard-32
192	11.56	0.1	0.1	11.76	6.13%	180.24	93.88%	1.638545455	n2-standard-48
256	12.84	0.1	0.1	13.04	5.09%	242.96	94.91%	2.208727273	n2-standard-64

Pods per Node (with memory limits)

Instance memory		1	2	4	8	16	64	128	192	256
Instance memory available for pods		0.55	1.3	2.8	6	13.2	58.32	118.48	180.24	242.96
Pod memory size	0.5	1	2	5	12	26	116	236	360	485
	1	0	1	2	6	13	58	118	180	242
	2	0	0	1	3	6	29	59	90	121
	4	0	0	0	1	3	14	29	45	60
	8	0	0	0	0	1	7	14	22	30
	16	0	0	0	0	0	3	7	11	15

Pods per Node (with memory limits) CAPPED

Instance memory		1	2	4	8	16	64	128	192	256
Pod memory size	0.5	1	2	5	12	26	110	110	110	110
	1	0	1	2	6	13	58	110	110	110
	2	0	0	1	3	6	29	59	90	110
	4	0	0	0	1	3	14	29	45	60
	8	0	0	0	0	1	7	14	22	30
	16	0	0	0	0	0	3	7	11	15

Pods per Node (with memory limits) Efficiency

Instance memory		1	2	4	8	16	64	128	192	256
Pod memory size	0.5	50.00%	50.00%	62.50%	75.00%	81.25%	85.94%	42.97%	28.65%	21.48%
	1	0.00%	50.00%	50.00%	75.00%	81.25%	90.63%	85.94%	57.29%	42.97%
	2	0.00%	0.00%	50.00%	75.00%	75.00%	90.63%	92.19%	93.75%	85.94%
	4	0.00%	0.00%	0.00%	50.00%	75.00%	87.50%	90.63%	93.75%	93.75%
	8	0.00%	0.00%	0.00%	0.00%	50.00%	87.50%	87.50%	91.67%	93.75%
	16	0.00%	0.00%	0.00%	0.00%	0.00%	75.00%	87.50%	91.67%	93.75%

CPU cores	CPU available	CPU quota per Pod (in millicores)
1	840	7.636363636
2	1830	16.63636364
4	3820	34.72727273
8	7810	71
16	15790	143.5454545
32	31750	288.6363636
64	63670	578.8181818

	Millicores reserved for the kubelet and kubernetes a					
CPU cores	EKS					
	Kubelet	OS	Total reserved	Total reserved %	Available	Available %
1	60	100	160	16.00%	840	84.00%
2	70	100	170	8.50%	1830	91.50%
4	80	100	180	4.50%	3820	95.50%
8	90	100	190	2.38%	7810	97.63%
16	110	100	210	1.31%	15790	98.69%
32	150	100	250	0.78%	31750	99.22%
48	190	100	290	0.60%	47710	99.40%
64	230	100	330	0.52%	63670	99.48%

Memory										
EKS	Instance type	Max Pods per no	Kubelet	OS	Hard eviction	Total Reserved	Total Reserved %	Available for worl	Available for worl	Max density
1	t2.micro	110	1.465	0.1	0.1	1.665	166.50%	0	0.00%	0
2	t3.small	110	1.465	0.1	0.1	1.665	83.25%	0.335	16.75%	0.003045454545
4	t3.medium	110	1.465	0.1	0.1	1.665	41.63%	2.335	58.38%	0.02122727273
8	m5.large	110	1.465	0.1	0.1	1.665	20.81%	6.335	79.19%	0.05759090909
16	m5.xlarge	110	1.465	0.1	0.1	1.665	10.41%	14.335	89.59%	0.1303181818
64	m5.4xlarge	110	1.465	0.1	0.1	1.665	2.60%	62.335	97.40%	0.5666818182
128	m5.8xlarge	250	3.005	0.1	0.1	3.205	2.50%	124.795	97.50%	0.49918
192	m5.12xlarge	250	3.005	0.1	0.1	3.205	1.67%	188.795	98.33%	0.75518
256	m5.16xlarge	250	3.005	0.1	0.1	3.205	1.25%	252.795	98.75%	1.01118

Pods per Node (with memory limits)

Instance memory		1	2	4	8	16	64	128	192	256
Instance memory available for pods		0	0.335	2.335	6.335	14.335	62.335	124.795	188.795	252.795
Pod memory size	0.5	0	0	4	12	28	124	249	377	505
	1	0	0	2	6	14	62	124	188	252
	2	0	0	1	3	7	31	62	94	126
	4	0	0	0	1	3	15	31	47	63
	8	0	0	0	0	1	7	15	23	31
	16	0	0	0	0	0	3	7	11	15

Pods per Node (with memory limits) CAPPED

Instance memory		1	2	4	8	16	64	128	192	256
Max pods per node		110	110	110	110	110	110	250	250	250
Pod memory size	0.5	0	0	4	12	28	110	249	250	250
	1	0	0	2	6	14	62	124	188	250
	2	0	0	1	3	7	31	62	94	126
	4	0	0	0	1	3	15	31	47	63
	8	0	0	0	0	1	7	15	23	31
	16	0	0	0	0	0	3	7	11	15

Pods per Node (with memory limits) Efficiency

Instance memory		1	2	4	8	16	64	128	192	256
Pod memory size	0.5	0.00%	0.00%	50.00%	75.00%	87.50%	85.94%	97.27%	65.10%	48.83%
	1	0.00%	0.00%	50.00%	75.00%	87.50%	96.88%	96.88%	97.92%	97.66%
	2	0.00%	0.00%	50.00%	75.00%	87.50%	96.88%	96.88%	97.92%	98.44%
	4	0.00%	0.00%	0.00%	50.00%	75.00%	93.75%	96.88%	97.92%	98.44%
	8	0.00%	0.00%	0.00%	0.00%	50.00%	87.50%	93.75%	95.83%	96.88%
	16	0.00%	0.00%	0.00%	0.00%	0.00%	75.00%	87.50%	91.67%	93.75%

CPU cores	Instance type	Max Pods per no	CPU available	CPU quota per Pod (in millicores)
1	t2.micro	4	840	210
2	m5.large	29	1830	63.10344828
4	m5.xlarge	58	3820	65.86206897
8	m5.2xlarge	58	7810	134.6551724
16	m5.4xlarge	234	15790	67.47863248
32	m5.8xlarge	234	31750	135.6837607
64	m5.16xlarge	737	47710	64.73541384

	Millicores reserved for the kubelet and kubernetes a									
CPU cores	AKS									
	Kubelet	OS	Total reserved	Total reserved %	Available	Available %				
1	60	100	160	16.00%	840	84.00%			Max pods per no	250
2	100	100	200	10.00%	1800	90.00%				
4	140	100	240	6.00%	3760	94.00%				
8	180	100	280	3.50%	7720	96.50%				
16	260	100	360	2.25%	15640	97.75%				
32	420	100	520	1.63%	31480	98.38%				
64	740	100	840	1.31%	63160	98.69%				

Memory									
AKS	Kubelet	OS	Hard eviction	Total Reserved	Total Reserved %	Available for worl	Available for worl	Max density	
1	0.25	0.1	0.75	1	100.00%	0	0.00%	0	Standard_B1s
2	0.5	0.1	0.75	1.35	67.50%	0.65	32.50%	0.0026	Standard_A1_v2
4	1	0.1	0.75	1.85	46.25%	2.15	53.75%	0.0086	Standard_A2_v2
8	1.8	0.1	0.75	2.65	33.13%	5.35	66.88%	0.0214	Standard_A4_v2
16	2.6	0.1	0.75	3.45	21.56%	12.55	78.44%	0.0502	Standard_A8_v2
64	5.48	0.1	0.75	6.33	9.89%	57.67	90.11%	0.23068	Standard_A8m_v2
128	9.32	0.1	0.75	10.17	7.95%	117.83	92.05%	0.47132	Standard_D32_v3
192	11.56	0.1	0.75	12.41	6.46%	179.59	93.54%	0.71836	Standard_D48_v3
256	12.84	0.1	0.75	13.69	5.35%	242.31	94.65%	0.96924	Standard_D64_v3

Pods per Node (with memory limits)

Instance memory		1	2	4	8	16	64	128	192	256
Instance memory available for pods		0	0.65	2.15	5.35	12.55	57.67	117.83	179.59	242.31
Pod memory size	0.5	0	1	4	10	25	115	235	359	484
	1	0	0	2	5	12	57	117	179	242
	2	0	0	1	2	6	28	58	89	121
	4	0	0	0	1	3	14	29	44	60
	8	0	0	0	0	1	7	14	22	30
	16	0	0	0	0	0	3	7	11	15

Pods per Node (with memory limits) CAPPED

Instance memory		1	2	4	8	16	64	128	192	256
Pod memory size	0.5	0	1	4	10	25	115	235	250	250
	1	0	0	2	5	12	57	117	179	242
	2	0	0	1	2	6	28	58	89	121
	4	0	0	0	1	3	14	29	44	60
	8	0	0	0	0	1	7	14	22	30
	16	0	0	0	0	0	3	7	11	15

Pods per Node (with memory limits) Efficiency

Instance memory		1	2	4	8	16	64	128	192	256
Pod memory size	0.5	0.00%	25.00%	50.00%	62.50%	78.13%	89.84%	91.80%	65.10%	48.83%
	1	0.00%	0.00%	50.00%	62.50%	75.00%	89.06%	91.41%	93.23%	94.53%
	2	0.00%	0.00%	50.00%	50.00%	75.00%	87.50%	90.63%	92.71%	94.53%
	4	0.00%	0.00%	0.00%	50.00%	75.00%	87.50%	90.63%	91.67%	93.75%
	8	0.00%	0.00%	0.00%	0.00%	50.00%	87.50%	87.50%	91.67%	93.75%
	16	0.00%	0.00%	0.00%	0.00%	0.00%	75.00%	87.50%	91.67%	93.75%

CPU cores	CPU available	CPU quota per Pod (in millicores)
1	840	3.36
2	1800	7.2
4	3760	15.04
8	7720	30.88
16	15640	62.56
32	31480	125.92
64	63160	252.64