

---

## kube-capacity



This is a simple CLI that provides an overview of the resource requests, limits, and utilization in a Kubernetes cluster. It attempts to combine the best parts of the output from `kubectl top` and `kubectl describe` into an easy to use CLI focused on cluster resources.

### Installation

Go binaries are automatically built with each release by GoReleaser. These can be accessed on the [GitHub releases page](#) for this project.

### Homebrew

This project can be installed with Homebrew:

```
1 brew tap robscott/tap
2 brew install robscott/tap/kube-capacity
```

### Krew

This project can be installed with Krew:

```
1 kubectl krew install resource-capacity
```

### Usage

By default, kube-capacity will output a list of nodes with the total CPU and Memory resource requests and limits for all the pods running on them. For clusters with more than one node, the first line will also include cluster wide totals. That output will look something like this:

```
1 kube-capacity
2
3 NODE           CPU REQUESTS   CPU LIMITS   MEMORY REQUESTS
4 *             560m (28%)    130m (7%)    572Mi (9%)      770
5               Mi (13%)
6 example-node-1 220m (22%)    10m (1%)    192Mi (6%)      360
7               Mi (12%)
```

---

6	example-node-2	340m (34%)	120m (12%)	380Mi (13%)	410
	Mi (14%)				

## Including Pods

For more detailed output, kube-capacity can include pods in the output. When `-p` or `--pods` are passed to kube-capacity, it will include pod specific output that looks like this:

```
1 kube-capacity --pods
2
3 NODE          NAMESPACE      POD          CPU REQUESTS
4 *            *              *
5             CPU LIMITS    MEMORY REQUESTS    MEMORY LIMITS
6 example-node-1 *              *              220m (22%)
7             320m (32%)    192Mi (6%)        360Mi (12%)
8 example-node-1 kube-system    metrics-server-lwc6z 100m (10%)
9             200m (20%)    100Mi (3%)        200Mi (7%)
10 example-node-1 kube-system    coredns-7b5bcb98f8 120m (12%)
11             120m (12%)    92Mi (3%)         160Mi (5%)
12
13 example-node-2 *              *              340m (34%)
14             460m (46%)    380Mi (13%)       410Mi (14%)
15 example-node-2 kube-system    kube-proxy-3ki7      200m (20%)
16             280m (28%)    210Mi (7%)        210Mi (7%)
17 example-node-2 tiller        tiller-deploy        140m (14%)
18             180m (18%)    170Mi (5%)        200Mi (7%)
```

## Including Utilization

To help understand how resource utilization compares to configured requests and limits, kube-capacity can include utilization metrics in the output. It's important to note that this output relies on metrics-server functioning correctly in your cluster. When `-u` or `--util` are passed to kube-capacity, it will include resource utilization information that looks like this:

```
1 kube-capacity --util
2
3 NODE          CPU REQUESTS    CPU LIMITS    CPU UTIL    MEMORY
4 *            REQUESTS      MEMORY LIMITS    MEMORY UTIL
5             560m (28%)    130m (7%)    40m (2%)    572Mi (9%)
6             770Mi (13%)    470Mi (8%)
7 example-node-1 220m (22%)    10m (1%)    10m (1%)    192Mi (6%)
8             360Mi (12%)    210Mi (7%)
9 example-node-2 340m (34%)    120m (12%)    30m (3%)    380Mi (13%)
10            410Mi (14%)    260Mi (9%)
```

---

## Displaying Available Resources

To more clearly see the total available resources on the node it is possible to pass the `--available` option to `kube-capacity`, which will give output in the following format

```
1 kube-capacity --available
2
3 NODE          CPU REQUESTS  CPU LIMITS  MEMORY REQUESTS
4 *             560/2000m    130/2000m    572/5923Mi
5 example-node-1 220/1000m    10/1000m    192/3200Mi
6 example-node-2 340/1000m    120/1000m    380/2923Mi
```

## Including Pods and Utilization

For more detailed output, `kube-capacity` can include both pods and resource utilization in the output. When `--util` and `--pods` are passed to `kube-capacity`, it will result in a wide output that looks like this:

```
1 kube-capacity --pods --util
2
3 NODE          NAMESPACE  POD          CPU REQUESTS
4 *             *          *            MEMORY REQUESTS
5              CPU LIMITS  CPU UTIL    MEMORY LIMITS  MEMORY
6              UTIL
7 example-node-1 *          *            560m (28%)
8              780m (38%)  340m (17%)  572Mi (9%)    770Mi (13%)  470Mi
9
10 example-node-1 *          *            220m (22%)
11              320m (32%)  160m (16%)  192Mi (6%)    360Mi (12%)  210Mi
12
13 example-node-1 kube-system metrics-server-lwc6z 100m (10%)
14              200m (20%)  70m (7%)    100Mi (3%)    200Mi (7%)    120Mi
15
16 example-node-1 kube-system coredns-7b5bcb98f8 120m (12%)
17              120m (12%)  90m (9%)    92Mi (3%)    160Mi (5%)    90Mi
18
19
20 example-node-2 *          *            340m (34%)
21              460m (46%)  180m (18%)  380Mi (13%)    410Mi (14%)  260Mi
22
23 example-node-2 kube-system kube-proxy-3ki7 200m (20%)
24              280m (28%)  110m (11%)  210Mi (7%)    210Mi (7%)    120Mi
25
26              (4%)
```

12	example-node-2	tiller	tiller-deploy	140m (14%)	
	180m (18%)	70m (7%)	170Mi (6%)	200Mi (7%)	140Mi
	(5%)				

It's worth noting that utilization numbers from pods will likely not add up to the total node utilization numbers. Unlike request and limit numbers where node and cluster level numbers represent a sum of pod values, node metrics come directly from metrics-server and will likely include other forms of resource utilization.

## Sorting

To highlight the nodes, pods, and containers with the highest metrics, you can sort by a variety of columns:

```
1 kube-capacity --util --sort cpu.util
2
3
```

	NODE	CPU REQUESTS	CPU LIMITS	CPU UTIL	MEMORY
	REQUESTS	MEMORY LIMITS	MEMORY UTIL		
4	*	560m (28%)	130m (7%)	40m (2%)	572Mi (9%)
	770Mi (13%)	470Mi (8%)			
5	example-node-2	340m (34%)	120m (12%)	30m (3%)	380Mi (13%)
	410Mi (14%)	260Mi (9%)			
6	example-node-1	220m (22%)	10m (1%)	10m (1%)	192Mi (6%)
	360Mi (12%)	210Mi (7%)			

**Note** Starting in v0.7.4 you can append `.percentage` to sort by percentage. For example, `kube-capacity --util --sort cpu.util.percentage`.

## Displaying Pod Count

To display the pod count of each node and the whole cluster, you can pass `--pod-count` argument:

```
1 $ kube-capacity --pod-count
2
3
```

	NODE	CPU REQUESTS	CPU LIMITS	MEMORY REQUESTS	MEMORY
	LIMITS	POD COUNT			
4	*	950m (2%)	200m (0%)	284Mi (0%)	284Mi (0%)
	10/220				
5	minikube	850m (5%)	100m (0%)	231Mi (1%)	231Mi (1%)
	8/110				
6	minikube-m02	100m (0%)	100m (0%)	53Mi (0%)	53Mi (0%)
	2/110				

---

## Filtering By Labels

For more advanced usage, kube-capacity also supports filtering by pod, namespace, and/or node labels. The following examples show how to use these filters:

```
1 kube-capacity --pod-labels app=nginx
2 kube-capacity --namespace default
3 kube-capacity --namespace-labels team=api
4 kube-capacity --node-labels kubernetes.io/role=node
```

## Filtering By Node Taints

Kube-capacity supports advanced filtering by taints. Users can filter in and filter out taints within the same expression. The following examples show how to use node taint filters:

```
1 kube-capacity --node-taints special=true:NoSchedule
2 kube-capacity --node-taints special:NoSchedule
```

These will return only special nodes.

```
1 kube-capacity --node-taints special=true:NoSchedule-
2 kube-capacity --node-taints special:NoSchedule-
```

These will filter out special nodes and return only unspecial nodes.

```
1 kube-capacity --node-taints special=true:NoSchedule,old-hardware:
  NoSchedule-
```

This will return special nodes that are not tainted with `old-hardware:NoSchedule`. In other words, display the special nodes but don't display the ones that are running on old hardware.

```
1 kube-capacity --no-taint
```

This will filter out all nodes with taints.

## JSON and YAML Output

By default, kube-capacity will provide output in a table format. To view this data in JSON or YAML format, the output flag can be used. Here are some sample commands:

```
1 kube-capacity --pods --output json
2 kube-capacity --pods --containers --util --output yaml
```

---

## CSV and TSV Output

If you would like the data in a comma or tab separated file to make importing the data into a spreadsheet easier the output flag has options for those as well. Here are some sample commands:

```
1 kube-capacity --pods --output csv
2 kube-capacity --pods --containers --util --output tsv
```

Note: the `--available` flag is ignored with these two choices as the values can be derived within a spreadsheet

## Flags Supported

1	<code>--as string</code>	user to impersonate command with
2	<code>--as-group string</code>	group to impersonate command with
3	<code>-c, --containers</code>	includes containers in output
4	<code>--context string</code>	context to use <b>for</b> Kubernetes config
5	<code>-h, --help</code>	help <b>for</b> kube-capacity
6	<code>--kubeconfig string</code>	kubeconfig file to use <b>for</b> Kubernetes
	<code>config</code>	
7	<code>-n, --namespace string</code>	only include pods from <b>this</b> namespace
8	<code>--namespace-labels string</code>	labels to filter namespaces with
9	<code>--no-taint</code>	exclude nodes with taints
10	<code>--node-labels string</code>	labels to filter nodes with
11	<code>-o, --output string</code>	output format <b>for</b> information
12		(supports: [table json yaml csv tsv ])
13		( <b>default</b> "table")
14	<code>-a, --available</code>	includes quantity available instead
	<code>of percentage used (ignored</code>	with csv or tsv output types)
15	<code>-t, --node-taints</code>	taints to filter nodes with
16	<code>-l, --pod-labels string</code>	labels to filter pods with
17	<code>-p, --pods</code>	includes pods in output
18	<code>--sort string</code>	attribute to sort results by (
	<code>supports:</code>	
19		[cpu.util cpu.request cpu.limit mem
		.util mem.request mem.limit cpu.
		util.percentage
20		cpu.request.percentage cpu.limit.
		percentage mem.util.percentage
		mem.request.percentage
21		mem.limit.percentage name])
22		( <b>default</b> "name")
23	<code>-u, --util</code>	includes resource utilization in
	<code>output</code>	
24	<code>--pod-count</code>	includes pod counts <b>for</b> each of the
	<code>nodes and the whole cluster</code>	

---

## Prerequisites

Any commands requesting cluster utilization are dependent on metrics-server running on your cluster. If it's not already installed, you can install it with the official helm chart.

## Similar Projects

There are already some great projects out there that have similar goals.

- kube-resource-report: generates HTML/CSS report for resource requests and limits across multiple clusters.
- kubectltop: a CLI similar to top for Kubernetes, focused on resource utilization (not requests and limits).

## Contributors

Although this project was originally developed by robscott, there have been some great contributions from others:

- endzyme
- justinbarrick
- Padarn
- nickatsegment
- fnickels
- isaacnboyd

## License

Apache License 2.0