Overscaler Documentation

Release alpha

Gleam AI

Mar 07, 2018

Contents

1 Overscaler **3**
- 1.1 How it works . . . 3
- 1.2 Usage . . . 3
- 1.3 Credits . . . 4

2 Installation **5**
- 2.1 From sources . . . 5

3 Labels **7**
- 3.1 Overscaler labels . . . 7
- 3.2 Metrics . . . 7
- 3.3 Rules . . . 9

4 overscaler **11**
- 4.1 overscaler package . . . 11

5 Indices and tables **17**

Python Module Index **19**

Contents:

CHAPTER 1

Overscaler

Stateful sets autoscaler for Google Kubernetes Engine.

- Documentation: https://overscaler.readthedocs.io.

1.1 How it works

Since Kubernetes lacks a autoscale system for Stateful Set pods, it is necessary to implement a new service to play this role. Overscaler may run externally or be deployed as a new Stateful Set within the cluster, in any case, permissions are required to access Kubernetes internal services.

Monitoring and autoscaling is based on Stateful Sets labels and each one should include a series of labels that define:

- Overscaler is On or Off for this Stateful Set
- Metrics that will be monitored.
- Rules that will be applied to rescale.

Periodically, Overscaler scans full cluster to obtain the Stateful Sets labels and, after checking them, starts monitoring each Pod.

During this monitoring, Overscaler realizes a set of GET requests to an internal Kubernete service called Heapster that returns metrics related to Pods status, and checks if any limit established by the rules is exceeded to rescale the respective Stateful Set.

1.2 Usage

1.2.1 Login and cluster credentials

The first step is to login with gcloud and get the cluster credentials to monitor. To login run:

```
$ gcloud auth login
```

Or if you prefer to log in with a service account:

```
$ gcloud auth activate-service-account --key-file /path/to/credentials.json
```

For more information about gcloud login with visit login

To get credentials run:

```
$ gcloud container clusters get-credentials CLUSTER_NAME --zone ZONE_NAME --project␣
↪PROJECT_NAME
```

1.2.2 Run Overscaler

Usage:

```
$ overscaler start [OPTIONS]
```

Start Overscaler to monitor and autoscale.

Monitoring and autoscaling are based on labels. Each Stateful Set must include a series of labels that define:

- Overscaler is On or Off for this Stateful Set.
- Metrics that will be monitored.
- Rules that will be applied to rescale.

Options:

-pr, --project TEXT Project name. [required]

-c, --cluster TEXT Cluster name. [required]

-z, --zone TEXT Project zone name [required]

-n, --namespace TEXT Cluster namespace, default to "default".

--refresh_cluster INTEGER Refresh period for cluster labels (seconds). Default to 600.

--refresh_statefulset INTEGER Refresh period for stateful set labels (seconds). Default to 300. (seconds).

--refresh_auth INTEGER Refresh period for Api authentication (seconds). Default to 300. (seconds).

--help Show this message and exit.

1.3 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

2.1 From sources

The sources for Overscaler can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git@github.com:GleamAI/overscaler.git
```

Or:

```
$ git clone https://github.com/GleamAI/overscaler.git
```

Once you have a copy of the source, you can install it with:

```
$ cd overscaler
$ make install
```

Or if you prefer to install with pip:

```
$ cd overscaler
$ make pip-install
```

If you don't have pip installed, this Python installation guide can guide you through the process.

CHAPTER 3

Labels

As already mentioned this system is based on labels to know what metrics to get and what rules to apply. This labels must be written in spec.template.metadata.labels within the deployment yaml file.

3.1 Overscaler labels

In addition to metrics and rules it is also necessary to add some extra labels for the correct operation of the system.

- app: Stateful Set name.
- overscaler: “true” or “false”, active or deactivate overscaler in this Stateful set.
- current-count: Rescaling counter. During monitoring, this value is reduced until 0, then is possible to rescale.
- autoscaler-count: Value to be assigned in “current-count” after rescaling.
- min-replicas: Maximum number of replicas for this stateful set.
- max-replicas: Minimum number of replicas for this stateful set.
- rescaling: Flag to know when a Stateful Set is being rescaled.

Current-count and autoscaler-count labels play a key role. Each type of service requires a certain time after start to configure and start working in parallel with the other replicas. With these labels we guarantee that time.

3.2 Metrics

Overscaler is designed for a customizable monitoring through labels, adding a label for each metric to monitor, and there are different sets of node and pod metrics.

Label format:

metric-n: “metric-name”

Example:

```
metric-1: "cpu-usage-percent"
```

However, it is still possible to monitor the entire node or pod using the label “all-metrics: true”.

3.2.1 Node metrics

These metrics determine the status of the different nodes and are assigned by labels in the Google Kubernetes Engine.

Table 3.1: Node metrics

Metric Name	Description
cpu-limit	Cpu hard limit in millicores.
cpu-node-capacity	Cpu capacity of a node.
cpu-node-allocatable	Cpu allocatable of a node.
cpu-node-reservation	Share of cpu that is reserved on the node allocatable.
cpu-node-utilization	Cpu utilization as a share of node allocatable.
cpu-request	Cpu request (the guaranteed amount of resources) in millicores.
cpu-usage	Cumulative cpu usage on all cores.
cpu-usage-rate	Cpu usage on all cores in millicores.
cpu-usage-percent	Cpu usage percent of total cpu Node.
memory-limit	Memory hard limit in bytes.
memory-major-page-faults	Number of major page faults.
memory-major-page-faults-rate	Number of major page faults per second.
memory-node-capacity	Memory capacity of a node.
memory-node-allocatable	Memory allocatable of a node.
memory-node-reservation	Share of memory that is reserved on the node allocatable.
memory-node-utilization	Memory utilization as a share of memory allocatable.
memory-page-faults	Number of page faults.
memory-page-faults-rate	Number of page faults per second.
memory-request	Memory request (the guaranteed amount of resources) in bytes.
memory-usage	Total memory usage.
memory-rss	RSS memory usage.
memory-working-set	Total working set usage. Working set is the memory being used and not easily dropped by the kerne
memory-usage-percent	Memory usage percent of total memory Node.
network-rx	Cumulative number of bytes received over the network.
network-rx-errors	Cumulative number of errors while receiving over the network.
network-rx-errors-rate	Number of errors while receiving over the network per second.
network-rx-rate	Number of bytes received over the network per second.
network-tx	Cumulative number of bytes sent over the network
network-tx-errors	Cumulative number of errors while sending over the network
network-tx-errors-rate	Number of errors while sending over the network
network-tx-rate	Number of bytes sent over the network per second.
uptime	Number of milliseconds since the container was started.

3.2.2 Pod metrics

These metrics determine the status of any Pods and are assigned by labels in the different Stateful sets.

Table 3.2: Pod metrics

Metric Name	Description
cpu-limit	Cpu hard limit in millicores.
cpu-request	Cpu request (the guaranteed amount of resources) in millicores.
cpu-usage-rate	Cpu usage on all cores in millicores.
cpu-usage-percent	Cpu usage percent of total node cpu.
memory-limit	Memory hard limit in bytes.
memory-major-page-faults-rate	Number of major page faults per second.
memory-page-faults-rate	Number of page faults per second.
memory-request	Memory request (the guaranteed amount of resources) in bytes.
memory-usage	Total memory usage.
memory-rss	RSS memory usage.
memory-working-set	Total working set usage. Working set is the memory being used and not easily dropped by the kernel.
memory-usage-percent	Memory usage percent of total node memory.
network-rx	Cumulative number of bytes received over the network.
network-rx-errors	Cumulative number of errors while receiving over the network.
network-rx-errors-rate	Number of errors while receiving over the network per second.
network-rx-rate	Number of bytes received over the network per second.
network-tx	Cumulative number of bytes sent over the network
network-tx-errors	Cumulative number of errors while sending over the network
network-tx-errors-rate	Number of errors while sending over the network
network-tx-rate	Number of bytes sent over the network per second.
uptime	Number of milliseconds since the container was started.

3.3 Rules

The rules for scaling are also assigned by labels and must have a specific syntax:

Label format:

rule-n: “metric_greater|lower_limit_scale|reduce”

- metric: Previously established metrics.
- greater or lower: “>” or “<” that limit.
- limit: Number that establishes a limit
- scale or reduce: Action to be realized when the limit is exceeded.

Example:

```
rule-1: "cpu-usage-percent_greater_90_scale"
rule-2: "memory-usage-percent_greater_90_scale"
rule-3: "cpu-usage-percent_lower_10_reduce"
rule-4: "memory-usage-percent_lower_10_reduce"
```

CHAPTER 4

overscaler

4.1 overscaler package

4.1.1 Submodules

4.1.2 overscaler.overcli module

4.1.3 overscaler.overprint module

overscaler.overprint.**print_cluster_info**(*autoscale*, *current_nodes*, *max_nodes*, *min_nodes*, *metrics*)

Prints Cluster information by console.

Parameters:

- **autoscale: bool** True if the node autoscale is active.
- **current_nodes: int** Number of current nodes.
- **max_nodes: int** Maximum number of allowed nodes.
- **min_nodes: int** Minimum number of allowed nodes.
- **metrics: array list** List of cluster metrics to monitor.

overscaler.overprint.**print_node_status**(*node_status*)

Prints Node status by console.

Parameters:

- **node_status: dict** Dictionary with all the information about the status of each node.

overscaler.overprint.**print_pod_status**(*pod_status*)

Prints Pod status by console.

Parameters:

- **pod_status: dict** Dictionary with all the information about the status of each pod.

overscaler.overprint.**print_statefulset_info**(*statefulset_labels*)
Prints Stateful Set information by console.

Parameters:

- **statefulset_lables: dict** Dictionary with metrics and rules of each stateful set.

4.1.4 overscaler.overtools module

overscaler.overtools.**actions**(*api*, *namespace*, *pod_status*, *statefulset_labels*, *max_nodes*)
Decision making based on pods status and stateful set rules.

Parameters:

- **api: pykube.http.HTTPClient** Http client for requests to Kubernetes Api.
- **namespace: str** Project namespace.
- **pod_status: dict** Dictionary with status pod information.
- **statefulset_lables: dict** Dict with metrics and rules of each stateful set.
- **max_nodes: int** Maximum number of allowed nodes.

overscaler.overtools.**check_rule**(*rule*, *typ*)
Checks the rules are well written.

Format rule: "metric_greater|lower_limit_scale|reduce"

Parameters:

- **rule: str** Rule to check.
- **type: str** Rule type, can be for node or pod

Returns:

- **check: bool** True if the rule has correct format.

overscaler.overtools.**get_cluster_labels**(*cluster_info*)
Gets cluster information.

Returns information about the number of nodes and their limits, node autoscale function and labels.

Parameters:

- **cluster_info: dict** Dictionary with all cluster information.

Returns:

- **autoscale: bool** True if node autoscale is active.
- **max_nodes: int** Maximum number of allowed nodes.
- **min_nodes: int** Minimum number of allowed nodes.
- **metrics: list** List of cluster metrics to monitor.

overscaler.overtools.**get_mean**(*metric*)
Calculates the arithmetic mean of a metric.

Parameters:

- **metric: dict** Dictionary with status metrics.

Returns:

- **mean: float** Arithmetic mean.

overscaler.overtools.**get_metrics**(*labels*, *typ*)

Get metrics from a dictionary of labels.

Parameters:

- **labels: dict** Dictionary with all metrics.
- **typ: str** Metrics type, "pod" or "cluster".

Returns:

- **metrics: str lst** List with metrics to monitor.

overscaler.overtools.**get_node_status**(*metrics*)

Gets Node status.

Returns information about state of all nodes.

Parameters:

- **metrics: str list** List of metrics to monitor.

Returns:

- **node_status: dict** Dictionary with all the information.

Returned dict format:

{

node_name1:{

metric-1: float, Metric-1 value.

metric-2: float, Metric-2 value.

... }

node_name2:{ ... }

... }

overscaler.overtools.**get_num_nodes**()

Returns number of active nodes.

Returns:

- **num_nodes: int** Number of current nodes.

overscaler.overtools.**get_pod_status**(*api*, *namespace*, *statefulset_labels*, *memory_allocatable*, *cpu_allocatable*)

Gets Pod status.

Returns information about state of all stateful set pods.

Parameters:

- **api: pykube.http.HTTPClient** Http client for requests to Kubernetes Api.
- **namespace: str** Project namespace.
- **statefulset_lables: dict** Dict with metrics for each stateful set.
- **memory_allocatable: int** Maximum memory allowed per node, expressed in bytes.
- **cpu_allocatable: int** Maximum memory allowed per node, expressed in minicores.

Returns:

- **pod_status: dict** Dictionary with all the information.

Returned dict format:

{

node_name1:{

pod-name1:{

metric-1: float, Metric-1 value.

metric-2: float, Metric-2 value.

... }

pod-name2:{ ... }

}

node_name2:{ ... }

... }

`overscaler.overtools.`**`get_rules`**(*labels*, *name*)

Get rules from a dictionary of labels.

Parameters:

- **labels: dict** Dictionary with all rules.
- **name: str** Stateful Set name.

Returns:

- **rules: str list** List with all rules to apply.

`overscaler.overtools.`**`get_statefulset_labels`**(*statefulset_info*)

Gets Stateful Set information. Returns information about labels, metrics and rules.

Parameters:

- **statefulset_info: dict** Dictionary with all Stateful Set information.

Returns:

- **statefulset_labels: dict** Dictionary with only the information needed for the overscaler.

Returned dict format:

{

statefulset_name1:{

overscaler: bool, Is overscaler active?

current-count:int, Autoscale pause counter.

autoscaler-count: int number, Number of waiting cycles after rescalling.

max-replicas: int, Maximum number of replicas.

min-replicas: int, Minimum number of replicas.

metrics: [str, str...], List with all metrics to monitor.

rules: [str,str...] List with all rules for this Stateful Set.

... }

statefulset_name2:{ ... }

... }

overscaler.overtools.**rescale**(*api*, *namespace*, *statefulset_name*, *action*, *max_nodes*)
Sets a new number of replicas for a given stateful set.

Parameters:

- **api: pykube.http.HTTPClient** Http client for requests to Kubernetes Api.
- **namespace: str** Project namespace.
- **statefulset_name: dict** Name of the statefulset to be rescaled.
- **action: str** Action to be realized. Can be "scale" o "reduce", one pods more or one pod less, respectively.
- **max_nodes: dict** Maximum number of allowed nodes.

overscaler.overtools.**start_proxy**()
Starts local proxy to Kubernetes cluster, host: 127.0.0.1:8001

overscaler.overtools.**update_current_count**(*api*, *namespace*, *statefulsets_labels*)
Updates the "current-count" label of all Stateful sets.

If its value is 0, this stateful set is ready to be scaled if is necessary.

Parameters:

- **api: pykube.http.HTTPClient** Http client for requests to Kubernetes Api.
- **namespace: str** Project namespace.
- **statefulset_lables: dict** Dict with metrics and rules of each stateful set.

4.1.5 Module contents

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

o

overscaler, 15
overscaler.overcli, 11
overscaler.overprint, 11
overscaler.overtools, 12

Index

A

actions() (in module overscaler.overtools), 12

C

check_rule() (in module overscaler.overtools), 12

G

get_cluster_labels() (in module overscaler.overtools), 12
get_mean() (in module overscaler.overtools), 12
get_metrics() (in module overscaler.overtools), 13
get_node_status() (in module overscaler.overtools), 13
get_num_nodes() (in module overscaler.overtools), 13
get_pod_status() (in module overscaler.overtools), 13
get_rules() (in module overscaler.overtools), 14
get_statefulset_labels() (in module overscaler.overtools), 14

O

overscaler (module), 15
overscaler.overcli (module), 11
overscaler.overprint (module), 11
overscaler.overtools (module), 12

P

print_cluster_info() (in module overscaler.overprint), 11
print_node_status() (in module overscaler.overprint), 11
print_pod_status() (in module overscaler.overprint), 11
print_statefulset_info() (in module overscaler.overprint), 11

R

rescale() (in module overscaler.overtools), 15

S

start_proxy() (in module overscaler.overtools), 15

U

update_current_count() (in module overscaler.overtools), 15