Resource Consumer

Overview

Resource Consumer is a tool which allows to generate cpu/memory utilization in a container. The reason why it was created is testing kubernetes autoscaling. Resource Consumer can help with autoscaling tests for: - cluster size autoscaling, - horizontal autoscaling of pod - changing the size of replication controller, - vertical autoscaling of pod - changing its resource limits.

Usage

Resource Consumer starts an HTTP server and handle sent requests. It listens on port given as a flag (default 8080). Action of consuming resources is send to the container by a POST http request. Each http request creates new process. Http request handler is in file resource_consumer_handler.go

The container consumes specified amount of resources:

- CPU in millicores,
- Memory in megabytes,
- Fake custom metrics.

Consume CPU http request

- suffix “ConsumeCPU”,
- parameters “millicores” and “durationSec”.

Consumes specified amount of millicores for durationSec seconds. Consume CPU uses “./consume-cpu/consume-cpu” binary (file consume-cpu/consume_cpu.go). When CPU consumption is too low this binary uses cpu by calculating math.sqrt(0) 10^7 times and if consumption is too high binary sleeps for 10 millisecond. One replica of Resource Consumer cannot consume more that 1 cpu.

Consume Memory http request

- suffix “ConsumeMem”,
- parameters “megabytes” and “durationSec”.

Consumes specified amount of megabytes for durationSec seconds. Consume Memory uses stress tool (stress -m 1 –vm-bytes megabytes –vm-hang 0 -t durationSec). Request leading to consuming more memory then container limit will be ignored.

Bump value of a fake custom metric

- suffix “BumpMetric”,
- parameters “metric”, “delta” and “durationSec”.

Bumps metric with given name by delta for durationSec seconds. Custom metrics in Prometheus format are exposed on "/metrics" endpoint.

CURL example

```
kubectl run resource-consumer --image=gcr.io/k8s-staging-e2e-test-images/resource-consumer:1
kubectl get services resource-consumer
```

There are two IPs. The first one is internal, while the second one is the external load-balanced IP. Both serve port 8080. (Use second one)

```
curl --data "millicores=300&durationSec=600" http://<EXTERNAL-IP>:8080/ConsumeCPU
```

300 millicores will be consumed for 600 seconds.

Image

Docker image of Resource Consumer can be found in Google Container Registry as gcr.io/k8s-staging-e2e-test-images/resource-consumer:1.9

Use cases

Cluster size autoscaling

1. Consume more resources on each node that is specified for autoscaler
2. Observe that cluster size increased

Horizontal autoscaling of pod

1. Create consuming RC and start consuming appropriate amount of resources
2. Observe that RC has been resized
3. Observe that usage on each replica decreased

Vertical autoscaling of pod

1. Create consuming pod and start consuming appropriate amount of resources
2. Observed that limits has been increased