Kubecost Turndown on GKE

Kubecost Turndown is an automated scaledown and scaleup of a Kubernetes cluster's backing node pool based on a custom schedule and criteria.

Setup

Service Account

In order to setup the scheduled turndown, you'll need to use a service account key with the following permissions:

- container.clusters.get
- container.clusters.update
- compute.instances.list
- iam.serviceAccounts.actAs
- container.nodes.create
- container.nodes.delete
- container.nodes.get
- container.nodes.getStatus
- container.nodes.list
- container.nodes.proxy
- container.nodes.update
- container.nodes.updateStatus

Included with the documentation is a `gke/create-service-key.sh` bash script which will:

- Create a new Role with the permissions above
- Create a new Service Account
- Assign the new Service Account the custom Role
- Generate a JSON Service Key `service-key.json`
- Use `kubectl` to create a kubernetes secret containing the service-key in the `kubecost` namespace

Note that in order to run `create-service-key.sh` successfully, you will need:

- Google Cloud, `gcloud` installed and authenticated.
- `kubectl` installed with target cluster in kubeconfig
    - `kubectl config current-context` should point to the target cluster before running the script.

The easiest way to use this script is to run:

```
$ ./gke/create-service-key.sh <Project ID> <Service Account Name>
```

The parameters to supply the script are as follows:

- **Project ID**: The GCP project identifier you can find via: `gcloud config get-value project`
- **Service Account Name**: The desired service account name to create

Note that if you have run this script more than once, the custom permissions role may have already been created. You may see an error similar to the following:

```
ERROR: (gcloud.iam.roles.create) Resource in project [PROJECT_ID] is the subject of a conflict: A role named kubecost.turndown in
projects/[PROJECT_ID] already exists.
```

This error is harmless, as the script should continue.

Kubectl Apply

In order to get the `kubecost-turndown` pod running on your cluster, use `kubectl` to apply the yaml descriptor `kubernetes/kubecost-turndown-pod.yaml`. This will create a the following for cluster specific access via the Kubernetes API:

- `ServiceAccount`
- `ClusterRole`
- `ClusterRoleBinding`
- `PersistentVolumeClaim`
- `Deployment`
- `Service`

```
$ kubectl apply -f kubernetes/kubecost-turndown.yaml -n kubecost
```

NOTE: It's important to apply using the `-n kubecost` namespace in order for the cluster role binding to work

Verify the Pod is Running

You can verify that the pod is running by issuing the following:

```
$ kubectl get pods -l app=kubecost-turndown -n kubecost
```

Kubecost Turndown Endpoints

To use the `kubecost-turndown` service, you will need to use a web request utility such as Postman or cURL.

Locate the Service Public IP

To locate the public IP to execute calls against, use the provided command:

```
$ kubectl get service kubecost-turndown-service -n kubecost

NAME                       TYPE           CLUSTER-IP   EXTERNAL-IP       PORT(S)
kubecost-turndown-service  LoadBalancer   10.0.10.80   130.211.127.184   9731:32265/TCP
```

When you execute HTTP requests, you will use the `EXTERNAL-IP` address. In the example above, `http://130.211.127.184:9731/`. This will be the URL base for all HTTP requests

Set a Turndown Schedule

To set a new schedule, send a `POST` to `/schedule` with the following body:

```
{
    "start": "Time",
    "end": "Time",
    "repeat": "none|daily|weekly"
}
```

- **start**: The starting date and time when turndown starts. RFC3339 Formatted.
- **end**: The date and time in which the cluster will turn back up. RFC3339 Formatted.
- **repeat**: There are currently three options for repeat. Note that this will apply a specific interval to turndown and turnup.

In the following example:

```
{
    "start": "2020-01-01T00:00:00.000Z",
    "end": "2020-01-01T12:00:00.000Z",
    "repeat": "daily"
}
```

Turndown will occur on Jan 1st at midnight UTC+0 and will turnup on Jan 1st at noon UTC+0. Because the repeat value is set to `daily`, this process will repeat itself on Jan 2nd, Jan 3rd, etc... at the same time. Note that overlapping schedules will return an error when scheduling.

The response will look something like this:

```
{
    "code": 200,
    "status": "success",
    "data": {
        "current": "scaledown",
        "scaleDownId": "cdf10707-0857-4ead-bcf9-4174ec35c4e3",
        "scaleDownTime": "2020-01-01T00:00:00Z",
        "scaleDownMetadata": {
            "repeat": "daily",
            "type": "scaledown"
        },
        "scaleUpID": "078891ff-4651-4348-ac85-fd747290847a",
        "scaleUpTime": "2020-01-01T12:00:00Z",
        "scaleUpMetadata": {
            "repeat": "daily",
            "type": "scaleup"
        }
    }
}
```

Helper Utility for Scheduling

There is a `set-schedule.sh` bash script provided with this documentation that can assist in setting a turndown schedule. It provides automation for the previous steps:

- Locates the service public IP
- Sends a request to the scheduling endpoint via `cURL`.

Make sure that your dates are properly RFC3339 formatted.

```
$ ./set-schedule.sh <START> <END> <REPEAT>
```

For example, to set the scheule for Jan 10, 2020 at 7:00 PM EST (UTC-5) to Jan 11, 9:00 AM EST (UTC-5) set to repeat daily, I would use the following:

```
$ ./set-schedule.sh "2020-01-10T19:00:00-05:00" "2020-01-11T09:00:00-05:00" "daily"
```

The Turndown Process

When the turndown schedule occurs, a new node pool with a single g1-small node is created. Taints are added to this node to only allow specific pods to be scheduled there. We update our kubecost-turndown deployment such that the turndown pod is allowed to schedule on the singleton node. Once the pod is moved to the new node, it will start back up and resume scaledown. This is done by cordoning all nodes in the cluster (other than our new g1-small node), and then reducing the node pool sizes to 0.

The Turnup Process

When it is time to turnup, we simply resize the nodes back to their original size.