Google Kubernetes Engine (GKE) (https://cloud.google.com/kubernetes-engine/) Documentation (https://cloud.google.com/kubernetes-engine/docs/) Guides

Scaling an application

This page explains how to scale a deployed application in Google Kubernetes Engine.

Overview

When you deploy an application (https://cloud.google.com/kubernetes-engine/docs/how-to/deploying-workloads-overview) in GKE, you define how many *replicas* of the application you'd like to run. When you *scale* an application, you increase or decrease the number of replicas.

Each replica of your application represents a Kubernetes Pod that encapsulates your application's container(s).

Before you begin

To prepare for this task, perform the following steps:

- Ensure that you have enabled the Google Kubernetes Engine API.

  ENABLE GOOGLE KUBERNETES ENGINE API (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/APIS/LIBRA

- Ensure that you have installed the Cloud SDK (https://cloud.google.com/sdk/downloads).
- Set your default project ID (https://support.google.com/cloud/answer/6158840):

  ```
  gcloud config set project [PROJECT_ID]
  ```

- If you are working with zonal clusters, set your default compute zone (https://cloud.google.com/compute/docs/zones#available):

  ```
  gcloud config set compute/zone [COMPUTE_ZONE]
  ```

- If you are working with regional clusters, set your default compute region (https://cloud.google.com/compute/docs/zones#available):

```
gcloud config set compute/region [COMPUTE_REGION]
```

- Update `gcloud` to the latest version:

```
gcloud components update
```

★ **Note:** You can override these default settings in `gcloud` commands using the `--project`, `--zone`, and `--region` operational flags.

Inspecting an application

Before scaling your application, you should inspect the application and ensure that it is healthy.

To see all applications deployed to your cluster, run `kubectl get [CONTROLLER]`. Substitute `[CONTROLLER]` for `deployments`, `statefulsets`, or another controller object type.

For example, if you run `kubectl get deployments` and you have created only one Deployment, the command's output should look similar to the following:

```
NAME                  DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
my-app                1         1         1            1           10m
```

The output of this command is similar for all objects, but may appear slightly different. For Deployments, the output has six columns:

- `NAME` lists the names of the Deployments in the cluster.
- `DESIRED` displays the desired number of *replicas*, or the *desired state*, of the application, which you define when you create the Deployment.
- `CURRENT` displays how many replicas are currently running.
- `UP-TO-DATE` displays the number of replicas that have been updated to achieve the desired state.
- `AVAILABLE` displays how many replicas of the application are available to your users.
- `AGE` displays the amount of time that the application has been running in the cluster.

In this example, there is only one Deployment, `my-app`, which has only one replica because its desired state is one replica. You define the desired state at the time of creation, and you can change it at any time by scaling the application.

Inspecting StatefulSets

Before scaling a StatefulSet, you should inspect it by running `kubectl describe statefulset my-app`.

In the output of this command, check the **Pods Status** field. If the `Failed` value is greater than `0`, scaling might fail.

If a StatefulSet appears to be unhealthy, run `kubectl get pods` to see which replicas are unhealthy. Then, run `kubectl delete [POD]`, where `[POD]` is the name of the unhealthy Pod.

Attempting to scale a StatefulSet while it is unhealthy may cause it to become unavailable.

Scaling an application

The following sections describe each method you can use to scale an application. The `kubectl scale` method is the fastest way to scale. However, you may prefer another method in some situations, like when updating configuration files or when performing in-place modifications.

KUBECTL SCALE | KUBECTL APPLY | CONSOLE

`kubectl scale` (https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#scale) lets your instantaneously change the number of replicas you want to run your application.

To use `kubectl scale`, you specify the new number of replicas by setting the `--replicas` flag. For example, to scale `my-app` to four replicas, run the following command, substituting ***[CONTROLLER]*** for `deployment`, `statefulset`, or another controller object type:

```
kubectl scale [CONTROLLER] my-app --replicas 4
```

If successful, this command's output should be similar to `deployment "my-app" scaled`.

Next, run `kubectl get [CONTROLLER] my-app`. The output should look similar to the following:

```
NAME                    DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
my-app                  4         4         4            4           15m
```

Note: It may take several minutes for scaling to complete.

Autoscaling Deployments

You can autoscale Deployments based on CPU utilization of Pods using `kubectl autoscale` or from the GKE Workloads menu in Cloud Console.

KUBECTL AUTOSCALE | CONSOLE

`kubectl autoscale` (https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#autoscale) creates a `HorizontalPodAutoscaler` (or HPA) object that targets a specified resource (called the *scale target*) and scales it as needed. The HPA periodically adjusts the number of replicas of the scale target to match the average CPU utilization that you specify.

Note: Your object's Pod template can include a `resources` field that specifies a CPU utilization request for each Pod. To learn more about making resource requests for Pods and containers, refer to Managing Compute Resources for Containers (https://kubernetes.io/docs/concepts/configuration/manage-compute-resources-container/)

When you use `kubectl autoscale`, you specify a maximum and minimum number of replicas for your application, as well as a CPU utilization target. For example, to set the maximum number of replicas to six and the minimum to four, with a CPU utilization target of 50% utilization, run the following command:

```
kubectl autoscale deployment my-app --max 6 --min 4 --cpu-percent 50
```

In this command, the `--max` flag is required. The `--cpu-percent` flag is the target CPU utilization over all the Pods. This command *does not* immediately scale the Deployment to six replicas, unless there is already a systemic demand.

After running `kubectl autoscale`, the `HorizontalPodAutoscaler` object is created and targets the application. When there a change in load, the object increases or decreases the application's replicas.

Note: The maximum number of replicas is limited by the cluster's resources. If the cluster has a static number of available nodes, the cluster might run out of resources and prevent some replicated Pods from running. To learn about scaling your cluster, refer to Resizing a Cluster

(https://cloud.google.com/kubernetes-engine/docs/how-to/resizing-a-container-cluster) and Cluster Autoscaling (https://cloud.google.com/kubernetes-engine/docs/concepts/cluster-autoscaler).

To see a specific `HorizontalPodAutoscaler` object in your cluster, run:

```
kubectl get hpa [HPA_NAME]
```

To see the `HorizontalPodAutoscaler` configuration:

```
kubectl get hpa [HPA_NAME] -o yaml
```

The output of this command is similar to the following:

```
apiVersion: v1
items:
- apiVersion: autoscaling/v1
  kind: HorizontalPodAutoscaler
  metadata:
    creationTimestamp: ...
    name: [HPA_NAME]
    namespace: default
    resourceVersion: "664"
    selfLink: ...
    uid: ...
  spec:
    maxReplicas: 10
    minReplicas: 1
    scaleTargetRef:
      apiVersion: apps/v1
      kind: Deployment
      name: [HPA_NAME]
    targetCPUUtilizationPercentage: 50
  status:
    currentReplicas: 0
    desiredReplicas: 0
kind: List
metadata: {}
resourceVersion: ""
selfLink: ""
```

In this example output, the `targetCPUUtilizationPercentage` field holds the `50` percentage value passed in from the `kubectl autoscale` example.

To see a detailed description of a specific `HorizontalPodAutoscaler` object in the cluster:

```
kubectl describe hpa [HPA_NAME]
```

You can modify the `HorizontalPodAutoscaler` by applying a new configuration file with `kubectl apply`, using `kubectl edit`, or using `kubectl patch`.

To delete a `HorizontalPodAutoscaler` object:

```
kubectl delete hpa [HPA_NAME]
```

Autoscaling with Custom Metrics

Beta

This product or feature is in a pre-release state and might change or have limited support. For more information, see the product launch stages (https://cloud.google.com/products/#product-launch-stages).

You can scale your Deployments based on custom metrics (https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/#support-for-custom-metrics) exported from [Stackdriver Kubernetes Engine Monitoring].

To learn how to use custom metrics to autoscale deployments, refer to the Autoscaling Deployments with Custom Metrics (https://cloud.google.com/kubernetes-engine/docs/tutorials/custom-metrics-autoscaling) tutorial.

Note: Stackdriver Kubernetes Engine Monitoring is a Google Cloud service separate from GKE. To use scaling based on custom metrics, you need to associate a paid Stackdriver service account with your Cloud Console project. For more information, refer to the Stackdriver Kubernetes Engine Monitoring documentation (https://cloud.google.com/monitoring/kubernetes-engine/).

What's next

- Learn about exposing your application externally (https://cloud.google.com/kubernetes-engine/docs/how-to/exposing-apps).

Last updated December 4, 2019.