Application Autoscaling

Objectives

- Configure a horizontal pod autoscaler for an application.

Kubernetes can autoscale a deployment based on current load on the application pods, by means of the `HorizontalPodAutoscaler` resource type.

Horizontal Pod Autoscaling

In a production environment, the load on an application can fluctuate significantly. Manually scaling the number of application pods to match demand is inefficient and prone to error. Red Hat OpenShift Container Platform (RHOCP) provides an automated solution for this challenge by using the Horizontal Pod Autoscaler (HPA). The HPA automatically scales the number of pods in a replication controller, deployment, replica set, or stateful set based on observed metrics such as CPU utilization.

For an enterprise, effective autoscaling is critical for several reasons:

- **High Availability and Performance**: By automatically adding pods during traffic spikes, the HPA ensures that the application remains responsive and meets its service-level objectives.

- **Cost Optimization**: During periods of low traffic, the HPA scales down the number of pods. Removing the extra pods releases unused resources back to the cluster, to enable other workloads to use them and to optimize overall resource usage.

- **Operational Efficiency**: Automating the scaling process removes the need for manual intervention by operations teams. The automation reduces the risk of human error and enables engineers to focus on other tasks.

The Autoscaling Process in Kubernetes

The HPA is implemented as a control loop that runs inside the Kubernetes controller manager. A `HorizontalPodAutoscaler` resource uses performance metrics that the RHOCP monitoring stack collects. The monitoring stack comes preinstalled in RHOCP. The autoscaler works in a continuous loop. By default, the controller manager checks HPA resources every 15 seconds. During each check, it performs the following steps:

- The autoscaler retrieves the target metric for scaling from the HPA resource definition.

- For each pod that the HPA resource targets, the autoscaler collects the current value of the metric from the metrics server.

- For each targeted pod, the autoscaler computes the usage percentage, based on the collected metric and the resource requests that are defined in the pod specification.

- The autoscaler computes the average usage across all the targeted pods.

- It establishes a ratio between the target metric value and the current average metric value.
- The autoscaler uses this ratio to make a scaling decision, either to increase or to decrease the number of replicas for the target resource.

Prerequisites for Autoscaling

To autoscale a deployment, you must specify resource requests for the containers in your pods. The HPA uses the requested resource values to calculate the utilization percentage. The autoscaler requires resource requests to calculate utilization and to perform scaling actions.

> **IMPORTANT**
>
> Pods that are created by using the `oc create deployment` command do not define resource requests by default. Using the RHOCP autoscaler might therefore require editing the deployment resources to add requests. Alternatively, you can create custom YAML resource files for your application. You can also add to your project a `LimitRange` resource, which defines default resource requests for pods.

Creating a Horizontal Pod Autoscaler

You can create an HPA resource by using the command line, a YAML manifest, or the web console.

Using the `oc autoscale` Command

To create a `HorizontalPodAutoscaler` resource from the command line, use the `oc autoscale` command:

```
[user@host ~]$ oc autoscale deployment/hello \
--min 1 --max 10 --cpu-percent 80
```

This command creates an HPA that targets the `hello` deployment. The HPA maintains the pod count between 1 and 10 replicas. It scales the deployment to keep the average CPU utilization of its pods at or below 80% of their requested CPU.

The maximum and minimum values for the horizontal pod autoscaler resource accommodate bursts of load and avoid overloading the RHOCP cluster. If the load on the application changes too quickly, then it might help to keep several spare pods to cope with sudden bursts of user requests. Conversely, too many pods can use up all cluster capacity and impact other applications that use the same RHOCP cluster.

Using a YAML Manifest

For more advanced configurations and for managing infrastructure as code, you can define an HPA resource in a YAML file.

```
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: hello
spec:
  minReplicas: 1    1
  maxReplicas: 10   2
  metrics: 3
  - resource:
      name: cpu
      target:
        averageUtilization: 80   4
        type: Utilization
    type: Resource
  scaleTargetRef:   5
    apiVersion: apps/v1
    kind: Deployment
    name: hello
```

1. The `minReplicas` field specifies the minimum number of pods.
2. The `maxReplicas` field specifies the maximum number of pods.
3. The `metrics` array defines the metrics to use for scaling decisions.
4. The `averageUtilization` field specifies the target average CPU utilization for each pod, and is represented as a percentage. If the global average CPU usage is above that value, then the horizontal pod autoscaler starts new pods. If the global average CPU usage is below that value, then the horizontal pod autoscaler deletes pods.
5. The `scaleTargetRef` field points to the resource that the HPA manages.

Use the `oc apply` command to create the resource from the file.

```
[user@host ~]$ oc apply -f hello-hpa.yaml
```

The preceding example creates an HPA resource that scales based on CPU usage. Alternatively, it can scale based on memory usage by setting the resource name to `memory`.

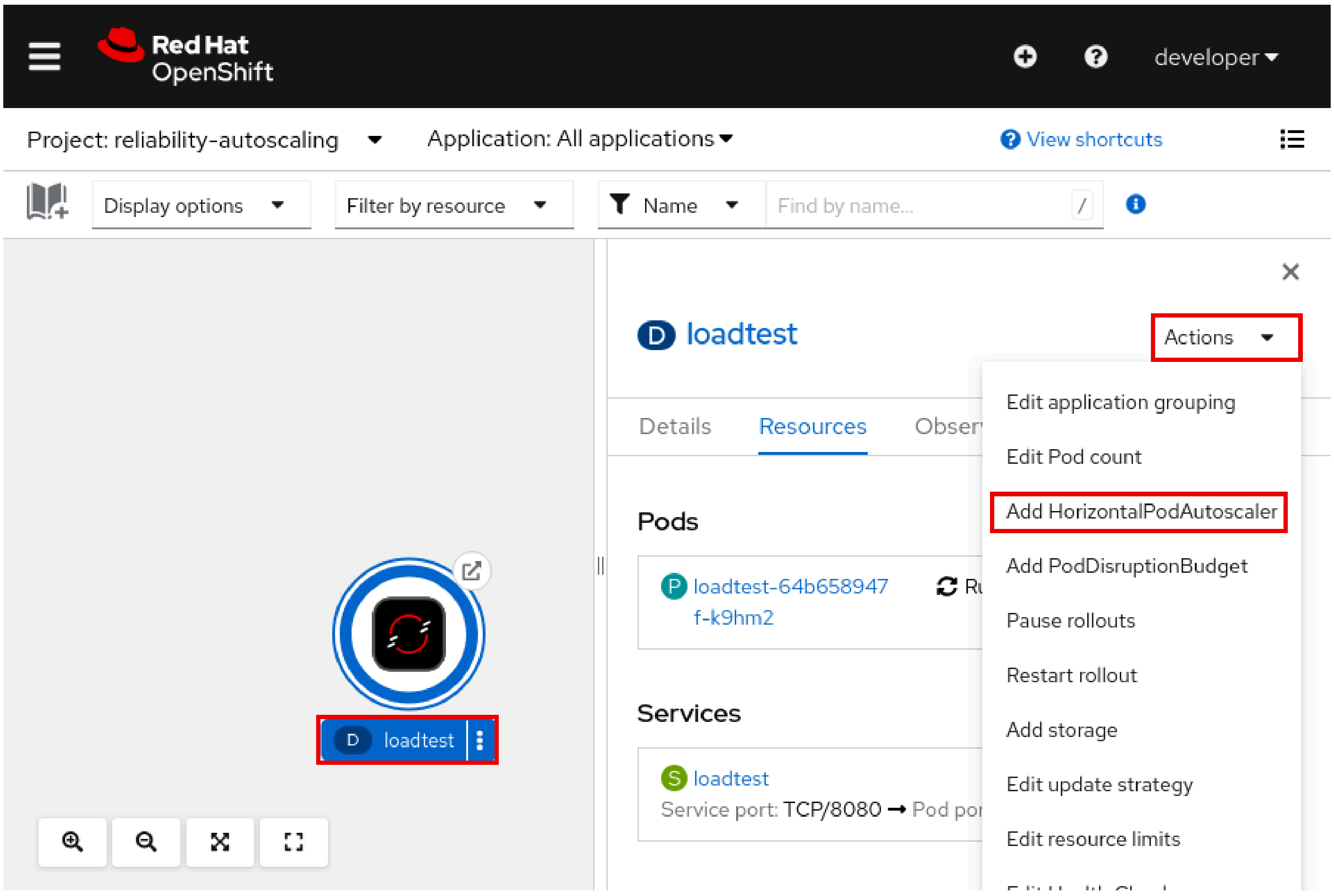
```
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: hello
spec:
  minReplicas: 1
  maxReplicas: 10
  metrics:
  - resource:
      name: memory   1
      target:
        averageUtilization: 80
...output omitted...
```

1 The resource name is set to `memory` to trigger scaling based on memory utilization.

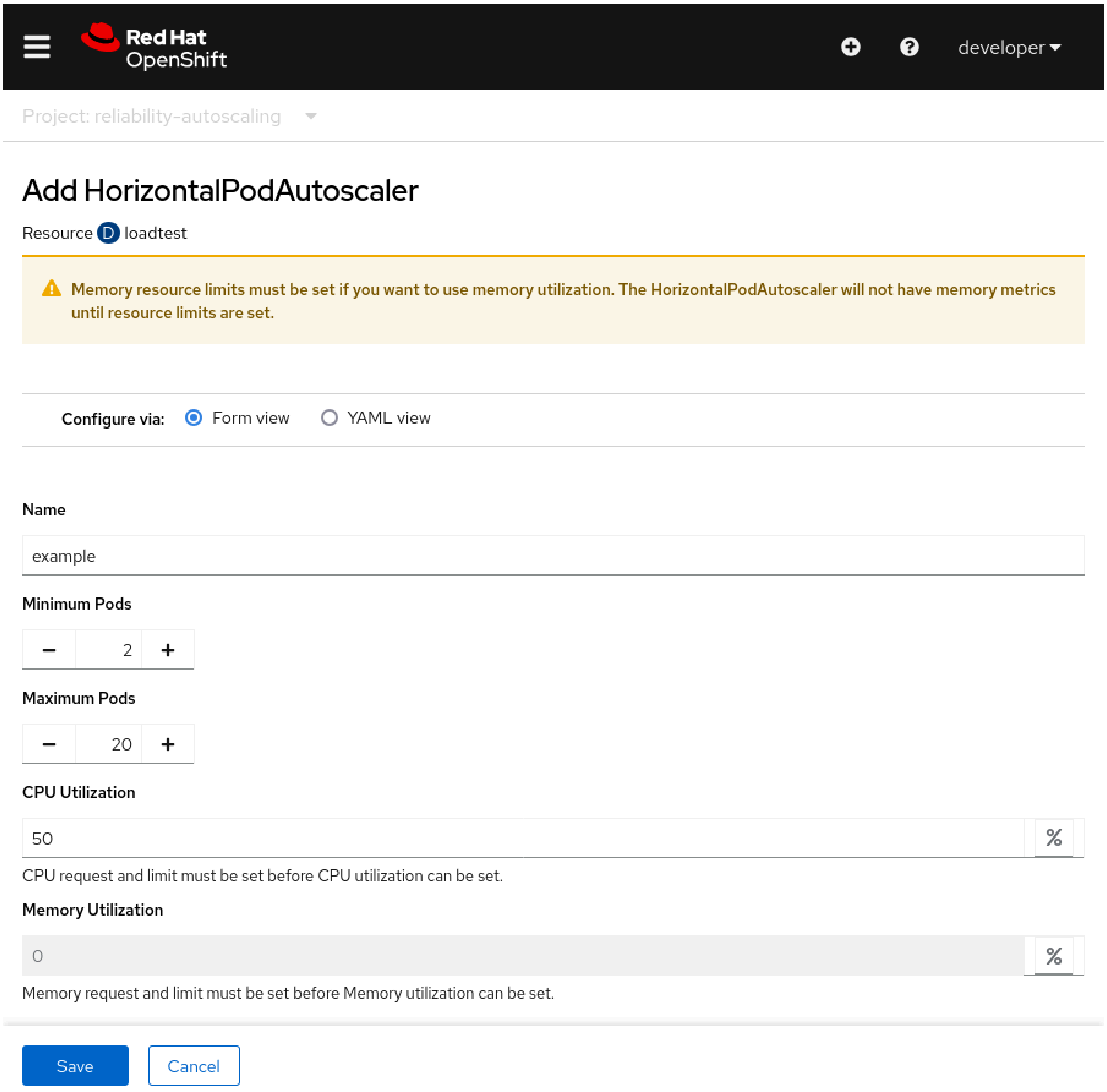
> **NOTE**
>
> Memory-based autoscaling is not suitable for all applications. For example, applications that are based on the Java Virtual Machine (JVM) often claim large memory at startup and do not release it. If an application's memory usage does not correlate with its load, then memory-based autoscaling might not be effective.

Using the Web Console

To create an HPA resource from the web console, in the `Developer` perspective, select the project and click the **Topology** menu. Click the `Deployment` for the application to display the details for the deployment. Select **Action** → **Add HorizontalPodAutoscaler** from the menu.



Use the `Add Horizontal Pod Autoscaler` form or YAML view to set the HPA parameters.



Add HorizontalPodAutoscaler

Resource D loadtest

Memory resource limits must be set if you want to use memory utilization. The HorizontalPodAutoscaler will not have memory metrics until resource limits are set.

Configure via: [✓ Form view] YAML view

Name: example

Minimum Pods: 2

Maximum Pods: 20

CPU Utilization: 50 %

CPU request and limit must be set before CPU utilization can be set.

Memory Utilization: 0 %

Memory request and limit must be set before Memory utilization can be set.

Verifying Horizontal Pod Autoscaler Status

After creating an HPA object, you can monitor its status and activity.

Viewing HPA Objects

To get information about HPA resources in the current project, use the `oc get hpa` command:

```
[user@host ~]$ oc get hpa
NAME    REFERENCE          TARGETS         MINPODS    MAXPODS    REPLICAS    AGE
hello   Deployment/hello   <unknown>/80%   1          10         1           30s
```

The horizontal pod autoscaler initially has the value of `<unknown>` in the `TARGETS` column. It might take up to five minutes for `<unknown>` to change and display a percentage for current usage. A persistent value of `<unknown>` in the `TARGETS` column might indicate that the target deployment does not define resource requests for the specified metric. The horizontal pod autoscaler scales only pods with resource requests.

Inspecting HPA Details and Events

For detailed information, use the `oc describe hpa` command. This command shows the HPA's configuration, current metrics, and a log of recent scaling events.

```
[user@host ~]$ oc describe hpa hello
...output omitted...
Events:
  Type    Reason             Age   From                         Message
  ----    ------             ----  ----                         -------
  Normal  SuccessfulRescale  16m   horizontal-pod-autoscaler  New size: 2; ...
  Normal  SuccessfulRescale  14m   horizontal-pod-autoscaler  New size: 4; ...
  Normal  SuccessfulRescale  14m   horizontal-pod-autoscaler  New size: 8; ...
  ...output omitted...
```

The `Events` section is useful for troubleshooting, by providing a chronological record of when and why the HPA scaled up or scaled down the target resource.

REFERENCES

For more information, refer to the *Automatically Scaling Pods with the Horizontal Pod Autoscaler* section in the *Working with Pods* chapter in the Red Hat OpenShift Container Platform 4.18 *Nodes* documentation at https://docs.redhat.com/en/documentation/openshift_container_platform/4.18/html-single/nodes/index#nodes-pods-autoscaling

For more information about creating horizontal pod autoscalers from the web console, refer to the *Creating a Horizontal Pod Autoscaler by Using the Web Console* section in the Red Hat OpenShift Container Platform 4.18 *Nodes* documentation at https://docs.redhat.com/en/documentation/openshift_container_platform/4.18/html-single/nodes/index#nodes-pods-autoscaling-creating-web-console_nodes-pods-autoscaling

For information about supported autoscaling metrics, refer to the *Understanding Horizontal Pod Autoscalers* section in the Red Hat OpenShift Container Platform 4.18 *Nodes* documentation at https://docs.redhat.com/en/documentation/openshift_container_platform/4.18/html-single/nodes/index#supported-metrics