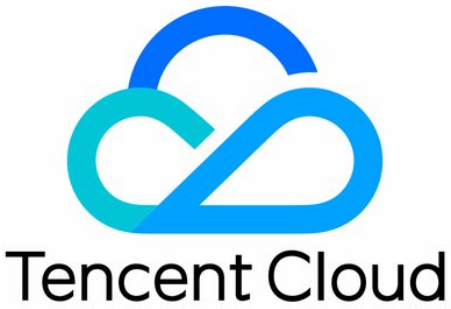
Tencent Kubernetes Engine

Elastic Cluster Guide

Product Documentation



Contents

Elastic Cluster Guide
Elastic Cluster Management
Creating Clusters

Last updated : 2020-11-27 11:07:38

Overview

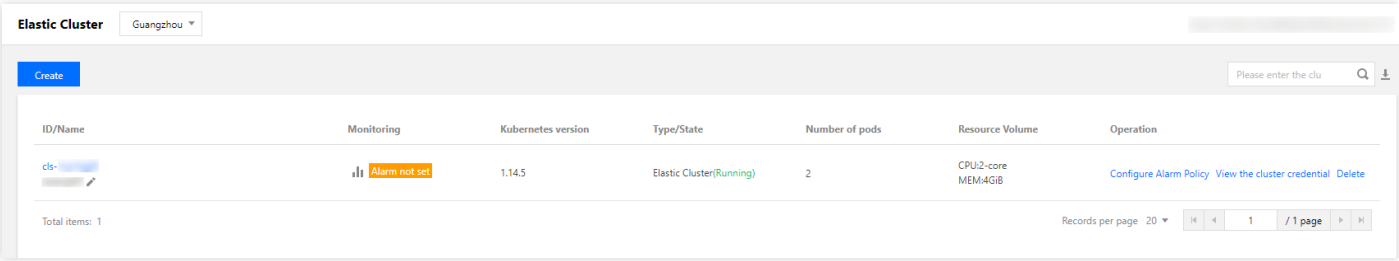
This document describes how to create an elastic cluster in the TKE console to help you get started with Elastic Kubernetes Service (EKS).

Prerequisites

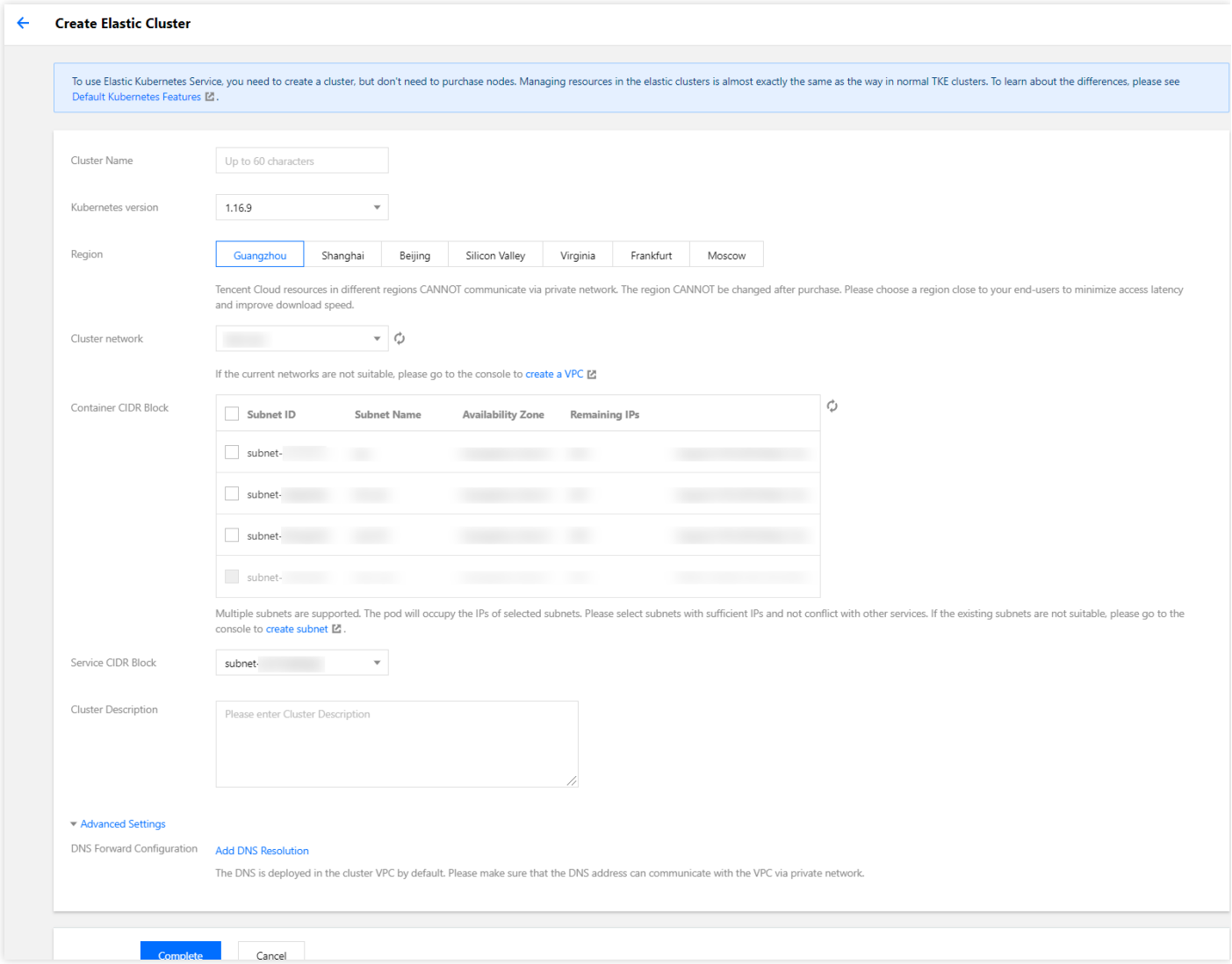
You have activated required permissions for EKS in the CAM console.

Directions

1. Log in to the TKE console and click **Elastic Cluster** in the left sidebar.
2. On the page that appears, select the region where you want to create the elastic cluster and then click **Create**, as shown in the following figure.



3. On the "Create Elastic Cluster" page that appears, specify the cluster information as instructed below:



- **Cluster Name**: the name of the cluster to be created. Up to 60 characters allowed.
- **Kubernetes Version**: the Kubernetes version of the cluster to be created. It supports V1.12 and later. For a comparison of the features of different versions, see Supported Versions of the Kubernetes Documentation.
- **Region**: the region where the cluster to be created is located. We recommend that you select the region closest to you to help minimize access latency and improve the download speed.
- **Cluster Network**: an existing VPC for elastic clusters. You can select a subnet of the VPC to serve as a container network for the elastic cluster. For more information, see VPC Documentation.
- **Container CIDR**: assigns an IP address for containers in the cluster. The IP addresses are in the IP address range of the container network. Pods in an elastic cluster directly use the IP addresses of a VPC subnet. We recommend that you select a subnet with sufficient available IP addresses that does not conflict with other products. For more information, see Container Network Description.
- **Service CIDR**: by default, the ClusterIP Service of the cluster is assigned in the selected VPC subnet. We recommend that you select a subnet with sufficient available IP addresses that does not conflict with other products.

- **Cluster Description**: specifies the information about the cluster to be created. This description will be displayed on the "Cluster Information" page.
- **DNS Forward Configuration**: this allows you to register upstream DNS information. You can enter a domain name to be resolved by the upstream DNS, for example, "example.org", and the upstream DNS address, for example, "10.0.0.1:53".

> ⚠ **Note :**
>
> The upstream DNS address must interconnect with the cluster network.

4. Click **Complete** to start creating the cluster. You can view the creation progress on the "Elastic Cluster" page.

Notes on the Container Network

In EKS, pods directly run on the specified VPC. Each pod is bound to an ENI in the specified VPC throughout its lifecycle. You can view the ENIs bound to the pods in the ENI List.

> ⓘ **Note :**
>
> - We recommend that you configure multiple availability zones for the container network so that your workloads can be automatically distributed to multiple availability zones, which improves usability.
> - Ensure that the subnet assigned to the container network has sufficient available IP addresses, so as to prevent pod creation failure caused by insufficient IPs when creating a large-scale workload.

Connecting to a Cluster

Last updated : 2020-04-27 11:41:29

Introduction

This document describes how to connect a local client to an elastic cluster through kubectl, which is the Kubernetes command-line tool.

Prerequisites

- Install cURL software program
- Select an appropriate way to obtain kubectl based on the OS type:

> Replace `v1.14.5` in the following commands with the actual version of your kubectl.

  - **MacOS**
    Run the following command to obtain kubectl:
    ```
    curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.14.5/bin/darwin/amd64/kubectl
    ```
  - **Linux**
    Run the following command to obtain kubectl:
    ```
    curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.14.5/bin/linux/amd64/kubectl
    ```
  - **Windows**
    Run the following command to obtain kubectl:
    ```
    curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.14.5/bin/windows/amd64/kubectl.exe
    ```

Directions

Installing kubectl

1. Install kubectl as instructed in Install and Set Up kubectl.

> - If you have already installed kubectl, skip this step.

- This step uses the Linux OS as an example.

2. Run the following command to grant permissions to use kubectl.

```
chmod +x ./kubectl
```

```
sudo mv ./kubectl /usr/local/bin/kubectl
```

3. Run the following command to verify the whether the installation is successful.

```
kubectl version
```

If the output is similar to the following version information, the installation was successful.

```
Client Version: version.Info{Major:"1", Minor:"14", GitVersion:"v1.14.5", GitCommit:"0e9fcb426b100a2ae
a5ed5c25b3d8cfbb01a8acf", GitTreeState:"clean", BuildDate:"2019-08-05T09:21:30Z", GoVersion:"go1.12.
5", Compiler:"gc", Platform:"windows/amd64"}
```

Obtaining the account, password, and certificate information of the cluster

1. Log in to the TKE console and click **Elastic Cluster** in the left sidebar.
2. On the **Elastic Cluster** page that appears, click the ID of the cluster that you want to connect to open the management page.
3. Click **Basic Information** in the left sidebar to go to the **Basic Information** page.
4. On the **Basic Information** page, click **Show Credential** for the **Cluster Credential** field.
5. In the **Cluster Credential** window that appears, do as follows depending on the actual situation:
   - Obtain the account, password, and certificate information and then click **Copy** or **Download** to save the CA certificate locally.
   - Enable **Internet Access** or **private network Access** and then connect to the cluster through the IP address.
     - **Intra-cluster access**: leave the values of **Public IP Access Address** and **Private Network Access Address** as their defaults, which indicate that public and private IP addresses are disabled. In this case, you can directly run kubectl commands on a server in the cluster without any configuration.
     - **Internet-based access**: set **Public IP Access Address** to **Enabled**. After that, you can gain access to the cluster through a public IP address. For more information, see Operating the cluster based on certificate information through kubectl.
     - **VPC-based access**: set **Private Network Access Address** to **Enabled**. You must specify the subnet of the VPC for the API server. In addition, ensure that the selected subset has an available IP address. After that, you can access the cluster through the private IP address. For more information, see Operating the cluster based on certificate information through kubectl.

Operating the cluster based on certificate information through kubectl

Sending a single kubectl operation request with certificate information

This method can be used to perform a single operation on a cluster. This way, you do not have to save the certificate information of the TKE cluster to the server for a long period of time.

Request:

Parameters in the kubectl command are described as follows:

```
-s "Domain information" --username=Username --password=Password --certificate-authority=Certificate path
```

- **Domain information**: indicates the obtained public or private IP address.
- **Username**: defaults to `admin` .
- **Password**: it is the same as that of `token` in the **Cluster Credential** window. You can obtain the token from Step 5.
- **Certificate path**: it is the same as that of `Cluster CA Certificate` in the **Cluster Credential** window. You can obtain the path from Step 5.

Examples

Run the following command to obtain the node information of a cluster:

```
kubectl get node -s "https://xxx.xx.xx.xxx:443/" --username=admin --password=6666o9oIB2gHD88882quIfLMy6666 --certificate-authority=/etc/kubernetes/cluster-ca.crt
```

Modifying the kubectl configuration file for long-term validity

This method can be used to perform long-term operations on a cluster through kubectl. You only need to configure the configuration file once, and the configuration file remains effective until it is modified.

1. Run the following example command to modify the password and certificate information in the kubectl configuration file.

   ```
   kubectl config set-credentials default-admin --username=admin --password=6666o9oIB2gHD88882quIfLMy6666
   kubectl config set-cluster default-cluster --server=https://xxx.xx.xx.xxx:443/ --certificate-authority=/etc/kubernetes/cluster-ca.crt
   kubectl config set-context default-system --cluster=default-cluster --user=default-admin
   kubectl config use-context default-system
   ```

2. After the configuration is completed, run the following example command to obtain the node information.

   ```
   kubectl get namespaces
   ```

   A message similar to the following is returned, indicating that the configuration was successful.

```
NAME STATUS AGE
default Active 11d
kube-system Active 11d
```

Enabling auto-complete for commands in kubectl

You can run the following command to enable auto-complete for all commands that you enter in kubectl, so as to improve the usability of kubectl.

```
source <(kubectl completion bash)
```

Unsupported Native Kubernetes Features

Last updated : 2020-11-30 10:49:56

Elastic cluster is a service of Elastic Kubernetes Service (EKS). Before you use this service, please read the following information.

Billing Method

EKS adopts the pay-as-you-go billing method. For more information about the bills, please see Billing Overview, Product Pricing, and Purchase Limits.

Pod Specification Configuration

Container runtime resources and bills depend on the pod specification configuration. Please note the pod specification configuration and specific methods supported by elastic clusters. For more information, please see Resource Specifications and Specifying Resource Specifications.

Pod Temporary Storage

Upon the creation of each pod, temporary image storage less than 20 GiB is allocated.

⚠ Note :

- Temporary image storage will be deleted when the pod lifecycle ends. Therefore, ensure that important data is not stored in it.
- Due to image storage, the actual available storage is less than 20 GiB.
- You are recommended to mount important data and large files to Volume for persistent storage.

Kubernetes Version

Kubernetes versions earlier than 1.12 are not supported.

Unsupported Features

Elastic clusters do not have nodes. Therefore, some dependent node components, such as Kubelet and Kube-proxy, are not supported.

Node

Currently, adding or managing a physical node is not supported.

Kernel

Only kernel parameters started with "net" can be defined.

Workloads

You cannot deploy workloads of the DaemonSet type through EKS.

Services

You cannot deploy services of the NodePort type through EKS.

Volumes

You cannot use Linux filesystem events with inotify, which is a feature of emptyDir volumes.

Kubernetes Object Management
Workload Management

Last updated : 2020-09-04 10:35:19

Scenario

This document describes how to select different types of workloads to run your services in an elastic cluster.

- Elastic clusters have no nodes. Workloads allocate resources for each pod based on parameter settings when they are created. For more information, see Specifying Resource Specifications.
- To create and manage Kubernetes objects by using a YAML file, specify annotations. For more information, see Annotation Description.

Prerequisites

- You have created an elastic cluster that is in Running state. For more information, see Creating a Cluster.
- The cluster has an appropriate namespace that is in Active state.

Workload Types

Deployment

A Deployment declares the pod template and pod running policies. It is used to deploy stateless applications. You can specify the number of replicas, scheduling policy, and update policy for pods running in the Deployment as needed.

StatefulSet

A StatefulSet is used to manage stateful applications. It creates a persistent identifier for each pod based on the specifications. The identifier will be retained after a pod is migrated, terminated, or restarted. When using persistent storage, you can map storage volumes to identifiers. If your application does not require any persistent identifier, we recommend that you use a Deployment to deploy the application.

Job

A Job creates one or more pods and ensures that these pods run according to specified rules until they are terminated. Jobs can be used in many scenarios, such as batch computing and data analysis. You can specify the number of repeated running, concurrency, and restart policy as required.

A Job maintains existing pods and will not create new pods after it is completed. You can view the logs of completed pods in "Logs". Deleting a Job removes the pods it created and the logs of these pods.

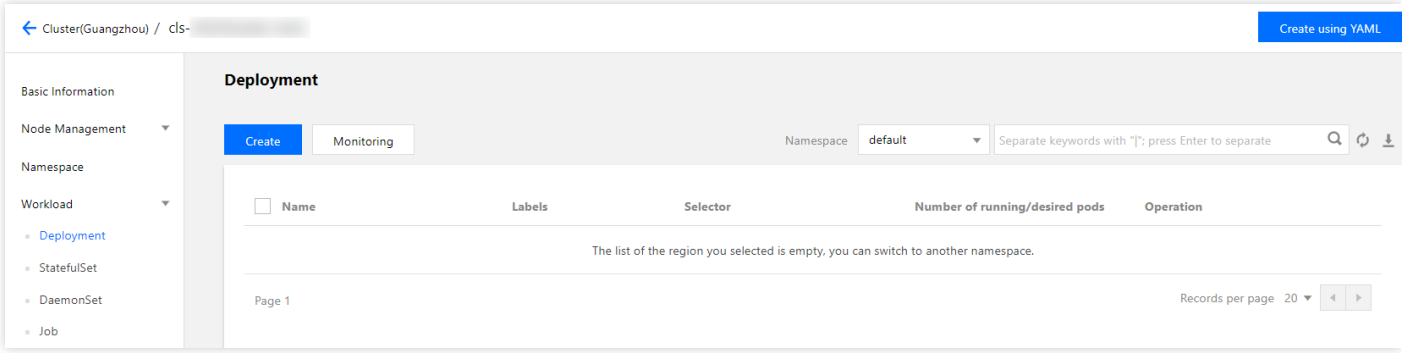
CronJob

A CronJob object is similar to a line in a crontab (cron table) file. It periodically runs a Job according to the specified schedule. For more information on the format, see Cron documentation.
The Cron format is as follows:

```
# File format description
# ——minute (0 - 59)
# | ——hour (0 - 23)
# | | ——day (1 - 31)
# | | | ——month (1 - 12)
# | | | | ——day of week (0 - 6)
# | | | | |
# * * * * *
```

Directions

1. Log in to the TKE console and click **Elastic Cluster** in the left sidebar.
2. On the "Elastic Cluster" page, click the ID of the cluster where the workload that you want to create resides. The **Deployment** page for the cluster appears, as shown in the following figure.



3. Click **Create** to go to the "Create a workload" page.
4. Specify the workload name and select a workload type.
   - For more information on parameter settings for each workload type, see the following:
     - Deployment Management
     - StatefulSet Management
     - CronJob Management
     - Job Management
   - For more information on how to perform other operations, see the following:
     - Setting the Resource Limit of a Workload
     - Setting the Scheduling Rule for a Workload

Specifying resource specifications

Last updated : 2020-09-04 10:34:27

Scenario

EKS specifies the maximum resources allocated to a pod by annotation specification or automatic Request and Limit calculation. You can select either method.

Directions

Annotation specification

EKS can add `template annotation` in the YAML file of a workload to explicitly specify pod resource specifications. For more information, see Annotation Description.

Automatic Request and Limit calculation

EKS can calculate the Request and Limit parameters set for a workload to determine the resources required for running pods. The calculation method varies depending on the pod resource type. For more information on how to automatically calculate specified resource specifications based on the Request and Limit parameters, see CPU specification calculation methods for pods and GPU specification calculation methods for pods.

- If `template annotation` is specified for a workload, the annotation configuration prevails and the Request and Limit parameters are not calculated.
- For more information on Request and Limit resource allocation, see the supported CPU and GPU specifications in Resource Specifications. If the set values vary greatly from the supported specifications, the allocated resources may exceed expectations, resulting in resource waste.
- Regardless of the set Request and Limit values, the final calculation result will be matched against Resource Specifications, and resources allocated to a pod will never exceed the allowed specifications.
- If Request and Limit are not set for a container in a pod, the Request and Limit values of the container are regarded as 0.
- If Request and Limit are not set for all containers in a pod, the default pod specifications are 1 CPU core and 2-GiB memory.
- Initcontainer and Container are calculated based on the following methods, and the larger value will be used.

CPU specifications calculation methods for pods

1. Calculate the total CPU and memory values of a pod.
   The total values are the greater **total Request value of all containers in a pod** and the greater **maximum Limit value of containers in a pod**, respectively.
2. Match pod resource specifications based on the following table.

Total CPU and Memory Values	Pod Resource Selection Rules
The total CPU and memory values are both 0.	The pod specifications are 1 CPU core and 2-GiB memory.
Either the total CPU value or the total memory value is 0.	Match the minimum value based on the non-0 total value. For example, if the total CPU value is 0 cores, and the total memory value is 8 GiB, match the minimum CPU value in the allowed specifications with 8-GiB memory. The selected pod specifications are 1 CPU core and 8-GiB memory.
Neither the total CPU value nor the total memory value is 0.	Match resource specifications in Resource Specifications. First, select a higher specification (specification A) with a CPU value that is equal to or similar to the total CPU value. Then, select a higher specification with a memory value that is similar to the total memory value. ◦ If the total memory value is less than the minimum memory value in the memory range of specification A, select the minimum memory value in the memory range of specification A. ◦ If the total memory value is greater than the maximum memory value in the memory range of specification A, select a higher specification (specification B) with a memory value similar to the total memory value and change the total CPU value to that of specification B. ◦ If the total memory value falls within the memory range of specification A, select the nearest greater dual-value.
Either the total CPU value or the memory value exceeds the maximum specification.	An error occurs, and resource matching fails.

You can better understand the CPU specification calculation methods for pods from the following examples.

- **Example 1**

```
resources:
limits:
cpu: "1"
memory: 2Gi
requests:
cpu: "1"
memory: 2Gi
```

- *Result**: the selected pod specification is 1 CPU core and 2-GiB memory.
- **Example 2**

```
## container1
resources:
limits:
cpu: "4"
memory: 4Gi
requests:
cpu: "2"
memory: 4Gi
## container2
resources:
limits:
cpu: "1"
memory: 2Gi
requests:
cpu: "1"
memory: 2Gi
```

  - Total CPU value: max((2+1),max(4,1)) = 4 cores
  - Total memory value: max((4+2),max(4,2)) = 6 GiB
  - *Result**: TKS does not support the pod specification of 4 CPU cores and 6-GiB memory, and 6 GiB is less than the minimum memory value in the specifications with 4 CPU cores. Therefore, adjust the minimum memory value in the specifications with 4 CPU cores. The selected pod specification is 4 CPU cores and 8-GiB memory.

GPU specification calculation methods for pods

- Typically, GPUs have the same `nvidia.com/gpu` parameter value as vGPUs, and the value must be an integer.
- vGPU can be regarded as an independent GPU type. For example, 1/4*V100 indicates that one fourth of the computing power of a V100 GPU card is virtualized to a complete card. During resource allocation, one GPU card is requested, that is, the nvidia.com/GPU value is 1.

1. Calculate the total GPU value of a pod.
   The total GPU value is the **total Request value of all containers in a pod**.
2. Match pod resource specifications based on the following table.

Total CPU, Memory, and GPU Values	Pod Resource Matching Rules
The total values must comply with specification requirements, for	First, select a higher specification (specification A) with a GPU value that is equal to or similar to the total GPU value. Then, calculate the CPU and memory values based on CPU specification calculation methods for pods to obtain the CPU specification (specification B).

example, 1, 2, 4, and 8.	◦ If the CPU and memory values of specification A are greater than or equal to those of specification B, select the GPU value of specification A. ◦ If the CPU and memory values of specification A are less than those of specification B, select a higher GPU specification (specification C) with the CPU and memory values similar to those of specification B. In this method, the allocated number of GPU cards are greater than that needed, and therefore should be avoided. To prevent waste, lower the requested CPU and memory values./**b>**
Any total value exceeds the maximum specifications.	An error occurs, and resource matching fails.

You can better understand the GPU specification calculation methods for pods from the following examples.

- **Example 1**

```
## eks.tke.cloud.tencent.com/gpu-type: V100
resources:
limits:
cpu: "8"
memory: 32Gi
nvidia.com/gpu: "1"
requests:
cpu: "4"
memory: 16Gi
nvidia.com/gpu: "1"
```

  ◦ Total GPU value: 1

  ii. Total CPU value: max(4,8) = 8 cores

  iii. Total memory value: max(16,32) = 32 GiB

  Result: 8 cores and 32 GiB are less than the CPU and memory values (8 cores and 40 GiB) of the V100 GPU specification (one card) in Resource Specifications. The ultimately selected pod specification is 8 CPU cores, 40-GiB memory, and 1x V100.

- **Example 2**

```
## eks.tke.cloud.tencent.com/gpu-type: V100
## container1
resources:
limits:
cpu: "8"
memory: 32Gi
nvidia.com/gpu: "1"
requests:
cpu: "4"
memory: 16Gi
nvidia.com/gpu: "1"
## container2
```

```
resources:
limits:
cpu: "32"
memory: 128Gi
nvidia.com/gpu: "1"
requests:
cpu: "16"
memory: 64Gi
nvidia.com/gpu: "1"
```

- Total GPU value: 1+1 = 2

ii. Total CPU value: max((4+16),max(8,32)) = 32 cores

iii. Toal memory value: max((16+64),max(32,128)) = 128 GiB

Result: 32 cores and 128 GiB are greater than the CPU and memory values (18 cores and 80 GiB) of the V100 GPU specification (two cards) but less than the CPU and memory values (36 cores and 160 GiB) of the V100 GPU specification (four cards).

The ultimately selected pod specification is 36 CPU cores, 160-GiB memory, and 4x V100, resulting in the waste of two GPU cards. In this case, the waste should be avoided.

Service Management

Last updated : 2020-10-28 14:30:55

Introduction

This document describes how to use Service and Ingress as entry points to expose workloads to external sources.

Prerequisites

- You have created an elastic cluster that is in the Running state. For more information, see Creating a cluster.
- The cluster has an appropriate namespace that is in the Active state.

Service Types

Service

Service defines policies for accessing backend Pods and provides a fixed virtual IP address for access. It also provides load balancing for all requests to Pods.

Service can be of the following types:

- Public network access: the public network access Service uses operates in Loadbalance and automatically creates a public network CLB instance. Public IP addresses can access backend Pods.
- Intra-cluster access: the intra-cluster access Service operates in ClusterIP mode and is used for access within the cluster.
- VPC private network access: the VPC private network access Service operates in Loadbalance and automatically creates a private network CLB instance. By using `annotations:service.kubernetes.io/qcloud-loadbalancer-internal-subnetid: subnet-xxxxxxxx` , you can use a private IP address from the VPC private network to access the backend Pod.

⚠ Note :

When the service type is public network access, `ClusterIP` of this service is disabled by default. You can add the following annotations in yaml to enable `ClusterIP` :

```
service.kubernetes.io/qcloud-clusterip-loadbalancer-subnetid: #Subnet ID of the service CIDR
```

Ingress

Ingress is a collection of rules that allow access to Services of a cluster. You can configure different forwarding rules to allow different URLs to access different Services.

In order for Ingress resources to operate properly, you must run `Ingress-controller`. TKE enables the CLB-based `l7-lb-controller` by default and supports HTTP, HTTPS, and nginx-ingress controllers. You can select Ingress controllers according to your needs.

Directions

For more information and instructions, see Service Management and Ingress Management.

Considerations

- Creating ClusterIP Service in an elastic cluster uses IP addresses from the Service CIDR. Make sure there are enough IP addresses in the subnet.
  In an elastic cluster, a CLB instance created by the Service binds all ENIs of all Pods within the endpoint.
- In an elastic cluster, Services only supports CLB instances.
- To create a Service using an existing CLB, you must ensure that the CLB is not bound to any listener.

Other Resource Management

Last updated : 2020-04-27 11:41:30

Introduction

This document describes how to manage other Kubernetes resources in the TKE console, including namespaces, configuration, and storage.

Prerequisites

An elastic cluster has been created and is in the Running state. For more information, see Creating a Cluster.

Directions

1. Log in to the TKE console and click **Elastic Cluster** in the left sidebar.
2. On the **Elastic Cluster** page that appears, click the ID of the desired cluster.
3. On the management page of the cluster, you can manage other resources according to the following documents:
   - To manage a namespace, see Namespace.
   - To manage Horizontal Pod Autoscaler (HPA), see HPA and HPA Metrics.
   - To manage configuration resources, see ConfigMap Management and Secret Management.
   - To manage storage resources, see Volume Management, PV and PVC Management, and StorageClass Management.

HPA Metrics

Last updated : 2020-05-25 09:34:49

Horizontal Pod Autoscaler (HPA) uses various metrics, such as target instance CPU usage, to adjust the number of Pods to automatically scale in or out. You can use the console to configure these metrics, including CPU, memory, disk, network, and GPU metrics. You can also use YAML files to create or modify HPAs. This article also contains a sample YAML file.

Metrics

The following table is a list of metrics used the HPA:

> Each variable in the `metricName` column has a Default unit. You can omit such units when editing the YAML file.

CPU metrics

Metric name (Console)	Unit (Console)	Description	Type	metricName	Default unit
CPU utilization	Cores	CPU utilization of the Pod	Pods	k8s_pod_cpu_core_used	Cores
CPU utilization (% of total resources)	%	The percentage of the CPU utilization out of all cores allocated to the Pod.	Pods	k8s_pod_rate_cpu_core_used_resource	%
CPU utilization (% of request)	%	The percentage of CPU utilization to the request value	Pods	k8s_pod_rate_cpu_core_used_request	%
CPU utilization	%	The percentage	Pods	k8s_pod_rate_cpu_core_used_limit	%

(% of limit)		of CPU utilization to the limit value			

Memory metrics

Metric name (Console)	Unit (Console)	Description	Type	metricName	Default unit
Memory usage	Mib	Pod memory usage	Pods	k8s_pod_mem_usage_bytes	B
Memory utilization (% of allocated resources)	%	The percentage of used memory out of all memory allocated to the Pod.	Pods	k8s_pod_rate_mem_usage_bytes_resource	%
Memory utilization (% of request)	%	The percentage of memory usage to the request value	Pods	k8s_pod_rate_mem_usage_request	%
Memory utilization (% of limit)	%	The percentage of memory usage to the limit value	Pods	k8s_pod_rate_mem_usage_limit	%

Using YAML files to Create and Modify HPAs

You can use YAML files to create and edit HPAs. The following is a sample YAML file. It creates an HPA called example which is triggered when CPU utilization reaches 1 and the number of Pods ranges from 1 to 2.

```
apiVersion: autoscaling/v2beta1
kind: HorizontalPodAutoscaler
metadata:
```

```
name: example
namespace: default
labels:
qcloud-app: example
spec:
minReplicas: 1
maxReplicas: 2
metrics:
- type: Pods
pods:
metricName: k8s_pod_cpu_core_used
targetAverageValue: "1"
scaleTargetRef:
apiVersion: apps/v1beta2
kind: Deployment
name: nginx
```

Annotation

Last updated : 2020-09-04 11:53:21

Workload Template Annotation Description

You can define `template annotation` in a YAML file to implement capabilities such as binding security groups and allocating resources for pods. For more information on the configuration method, see the following table.

⚠ Note :

- If no security group is specified, a pod is bound to the `default` security group in the same region by default. Ensure that the network policy of the `default` security group does not affect the pod.
- To allocate GPU resources, you must specify `eks.tke.cloud.tencent.com/gpu-type` .
- Except `eks.tke.cloud.tencent.com/gpu-type` , the other four annotations related to resource allocation in the following table are optional. If you specify them, ensure that they are correct.
- To allocate CPU resources, you must specify both `cpu` and `mem` and make sure that their values meet the CPU specifications in Resource Specifications. In addition, you can select Intel or AMD CPUs to allocate by specifying `cpu-type` . AMD CPUs are more cost-effective. For more information, see Pricing.
- To allocate GPU resources, you must specify the `cpu` , `mem` , `gpu-type` , and `gpu-count` annotations and ensure that their values meet the GPU specifications in Resource Specifications.

Annotation Key	Annotation Value and Description	Required
eks.tke.cloud.tencent.com/security-group-id	Default security group bound to a workload. Specify the security group ID. • Multiple security group IDs can be specified and separated by a comma (`,`), such as `sg-id1,sg-id2` . • Network policies take effect based on the sequence of security groups.	No. If you do not specify it, the `default` security group in the same region bound to the workload is associated by default. If you specify it, ensure that the security group ID already exists in the region where the workload resides.

Annotation Key	Annotation Value and Description	Required
eks.tke.cloud.tencent.com/cpu	Number of CPU cores required by a pod. For more information, see Resource Specifications. The unit is core by default.	No. If you specify it, ensure that the specifications are supported and specify the `cpu` and `mem` parameters.
eks.tke.cloud.tencent.com/mem	Memory required by a pod. For more information, see Resource Specifications. The unit must be included in the value, for example, 512 MiB, 0.5 GiB, or 1 GiB.	No. If you specify it, ensure that the specifications are supported and specify the `cpu` and `mem` parameters.
eks.tke.cloud.tencent.com/cpu-type	Model of the CPU resources required by a pod. Currently, supported models include: • intel • amd For more information on configurations supported by different models, see Resource Specifications.	No. If you do not specify it, the CPU type is not specified forcibly by default. The system will calculate the most suitable specifications according to Methods for Specifying Resource Specifications. If the calculated specifications are supported by both Intel and AMD, Intel CPUs are preferred.
eks.tke.cloud.tencent.com/gpu-type	Model of the GPU resources required by a pod. Currently, supported models include: • 1/4*V100 • 1/2*V100 • V100 • 1/4*T4 • 1/2*T4 • T4 For more information on configurations supported by different models, see Resource Specifications.	If GPU resources are required, this option is required. When specifying it, ensure that the GPU model is supported. Otherwise, an error will be reported.

Annotation Key	Annotation Value and Description	Required
eks.tke.cloud.tencent.com/gpu-count	Number of GPUs required by a pod. For more information, please see Resource Specifications. The unit is card by default.	No. If you specify it, ensure that the specifications are supported.
eks.tke.cloud.tencent.com/static-ip	Fixed IP address for a pod. You can enable this feature by setting the value to `true`. If this feature is enabled, the IP address of a StatefulSet or Bare pod will not change when the pod is updated or restarted.	No. This annotation is valid only for StatefulSet and Pod workloads.
eks.tke.cloud.tencent.com/role-name	CAM role associated to a pod. The value is a CAM role name. A pod can obtain the permission policy of the associated CAM role to facilitate cloud resource operations such as purchasing resources and reading from or writing to storage.	No. If you specify it, ensure that the CAM role exists.
eks.tke.cloud.tencent.com/monitor_port	Sets an open port for monitoring data for a pod, to facilitate collection by Prometheus and other components.	No. If you do not specify it, the default value is 9100.
eks.tke.cloud.tencent.com/custom_metrics_url	Sets the custom monitoring metric pull address for a pod. The monitoring data opened at this address will be automatically read and reported by the monitoring component.	No. If you specify it, please ensure that the opened data protocol can be recognized by the monitoring system, such as the Prometheus protocol and cloud monitoring data protocol.

Sample

The following example shows the complete GPU specifications of the security group bound to a pod.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
generation: 1
labels:
k8s-app: nginx
qcloud-app: nginx
name: nginx
namespace: default
spec:
progressDeadlineSeconds: 600
replicas: 1
revisionHistoryLimit: 10
selector:
matchLabels:
k8s-app: nginx
qcloud-app: nginx
strategy:
rollingUpdate:
maxSurge: 1
maxUnavailable: 0
type: RollingUpdate
template:
metadata:
annotations:
eks.tke.cloud.tencent.com/cpu: "2"
eks.tke.cloud.tencent.com/gpu-count: "1"
eks.tke.cloud.tencent.com/gpu-type: 1/4*V100
eks.tke.cloud.tencent.com/mem: 10Gi
eks.tke.cloud.tencent.com/security-group-id: "sg-dxxxxxx5,sg-zxxxxxxu"
eks.tke.cloud.tencent.com/static-ip: "true"
eks.tke.cloud.tencent.com/role-name: "cam-role-name"
eks.tke.cloud.tencent.com/monitor_port: "9123"
eks.tke.cloud.tencent.com/custom_metrics_url: "http://localhost:8080/metrics"
creationTimestamp: null
labels:
k8s-app: nginx
qcloud-app: nginx
spec:
containers:
- image: nginx:latest
imagePullPolicy: Always
name: nginx
resources:
limits:
cpu: "1"
memory: 2Gi
nvidia.com/gpu: "1"
requests:
```

```
cpu: "1"
memory: 2Gi
nvidia.com/gpu: "1"
terminationMessagePath: /dev/termination-log
terminationMessagePolicy: File
dnsPolicy: ClusterFirst
imagePullSecrets:
- name: qcloudregistrykey
restartPolicy: Always
schedulerName: default-scheduler
securityContext: {}
terminationGracePeriodSeconds: 30
```

EKS Annotation Description

Elastic Kubernetes Service (EKS) allows you to use existing CLBs to create services accessed through the public or private network. If you want to provide idle CLBs to created services or use the same CLB in a cluster, you can add annotations.

⚠ Note :

- Ensure that your EKS does not share the same CLB with the CVM.
- When existing CLBs are used:
  - Only CLBs created through the CLB console can be used. You cannot reuse CLBs automatically created by Tencent Kubernetes Engine (TKE).
  - Ports of services that share the same existing CLB cannot be the same.
  - Cross-cluster services cannot share the same CLB.

Sample

```
apiVersion: v1
kind: Service
metadata:
annotations:
service.kubernetes.io/tke-existed-lbid: lb-pxxxxxxq
name: servicename
namespace: default
spec:
externalTrafficPolicy: Cluster
ports:
- name: tcp-80-80
nodePort: 31728
port: 80
protocol: TCP
targetPort: 80
```

```
sessionAffinity: None
type: LoadBalancer
```

OPS Center
Monitoring and alarm

Last updated : 2020-04-28 18:47:39

Overview

Elastic Kubernetes Service (EKS) collects and displays metrics for clusters, workloads, pods, and containers.

Prerequisites

An elastic cluster has been created and is in the Running state. For more information, see Creating a Cluster.

Procedure

1. Log in to the TKE console and click **Elastic Cluster** in the left sidebar.
2. On the **Elastic Cluster** page that appears, click the ID of the cluster that you want to manage.
3. On the cluster resource management page, view the metrics and configure alarms for them. For more information, see the following documents:
   - Viewing Metrics
   - Setting Alarms

Monitoring and Alarm Metrics

Monitoring metrics

TKE currently provides monitoring metrics in the following dimensions. All metrics are **average values** of the statistics collected within the statistical period.

- For monitoring metrics for persistent volumes (PVs) used by a workload, see Block Storage and Cloud File Storage.
- For network monitoring metrics for a CLB associated with a Service, see the Cloud Load Balancer.

- For more information on how to create an alarm policy, see Creating an Alarm Policy.

Monitoring metrics for a cluster

Metric	Unit	Description
CPU Usage	Core	Total number of CPU cores used by the running pods in the cluster
MEM Usage	B	Total amount of memory used by the running pods in the cluster

Monitoring metrics for a workload

Metric	Unit	Description
Restart of Pods	Times	Total number of times the pods in the workload are restarted
CPU Usage	Core	Total number of CPU cores used by the pods in the workload
CPU Utilization (% of Pod Specification)	%	Percentage of the CPU cores allocated to the pods in the workload that is used by the pods
MEM Usage	B	Total amount of memory used by the running pods in the workload
MEM Utilization (% of Pod Specification)	%	Percentage of the memory capacity allocated to the pods in the workload that is used by the pods

Monitoring metrics for a pod

Metric	Unit	Description
Exception	-	Status of the pod, which can be normal or abnormal
CPU Usage	Core	Number of CPU cores used by the pod
CPU Utilization (% of Request)	%	Percentage of the total number of CPU cores specified by Request that is used by the pod
CPU Utilization (% of Limit)	%	Percentage of the total number of CPU cores specified by Limit that is used by the pod
CPU Utilization (% of Pod Specification)	%	Percentage of the CPU cores allocated to the pod that is used by the pod
MEM Usage	B	Amount of memory used by the pod, including the cache usage for the pod
MEM Utilization (% of Request)	%	Percentage of the total amount of memory specified by Request that is used by the pod
MEM Utilization (% of Limit)	%	Percentage of the total amount of memory specified by Limit that is used by the pod

Metric	Unit	Description
MEM Utilization (% of Pod Specification)	%	Percentage of the memory capacity allocated to the pod that is used by the pod

Monitoring metrics for a container

Metric	Unit	Description
CPU Usage	Core	Number of CPU cores used by the container
CPU Utilization (% of Request)	%	Percentage of the total number of CPU cores specified by Request that is used by the container
CPU Utilization (% of Limit)	%	Percentage of the total number of CPU cores specified by Limit that is used by the container
MEM Usage	B	Amount of memory used by the container, including the cache usage for the container
MEM Utilization (% of Request)	%	Percentage of the total amount of memory specified by Request that is used by the container
MEM Utilization (% of Limit)	%	Percentage of the total amount of memory specified by Limit that is used by the container

Alarm Metrics

TKE currently provides alarm metrics in the following dimensions. All metrics are **average values** of the statistics collected within the statistical period.

Alarm metrics for a pod

Metric	Unit	Description
CPU Utilization (% of Pod Specification)	%	Percentage of the CPU cores allocated to the pod that is used by the pod
MEM Utilization (% of Pod Specification)	%	Percentage of the memory capacity allocated to the pod that is used by the pod
CPU Utilization (% of Request)	%	Percentage of the total number of CPU cores specified by Request that is used by the pod
MEM Utilization (% of Request)	%	Percentage of the total amount of memory specified by Request that is used by the pod

Metric	Unit	Description
CPU Utilization (% of Limit)	%	Percentage of the total number of CPU cores specified by Limit that is used by the pod
MEM Utilization (% of Limit)	%	Percentage of the total amount of memory specified by Limit that is used by the pod
Restart of Pods	Times	Number of times the pod is restarted.
Pod Ready	-	Status of the pod. By default, an alarm is generated when the value is False.
CPU Usage	Core	Number of CPU cores used by the pod
MEM Usage	MB	Amount of memory used by the pod, including the cache usage for the pod