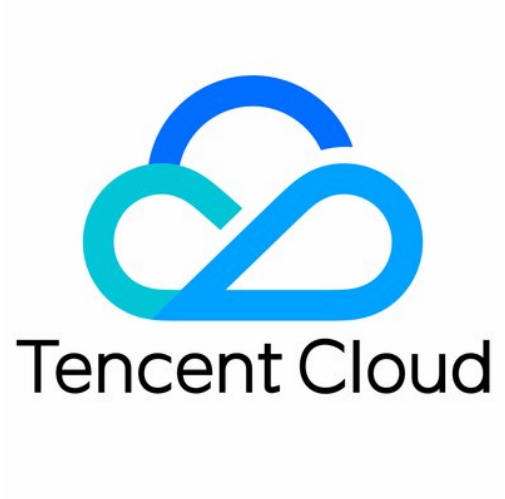
Cloud File Storage

Best Practices

Product Documentation



Contents

Best Practices
Using CFS on TKE

Last updated：2022-10-12 15:59:56

Overview

CFS is suitable for two use cases in a container environment:

Use case 1. Persistent storage of Pod/container data (dynamic mounting of CFS recommended)

CFS provides a space for persistent storage where the data is still stored when a Pod or container is terminated. In this way, the original space can be quickly mounted through PVCs to implement data reads/writes quickly when another Pod or container is started.
Compared with other schemes, a single-FS instance allows you to store the data from multiple Pods or containers and assign different subdirectories in the CFS instance to different Pods. Standard and High-Performance CFS instances are pay-as-you-go with no requirement for the minimum purchase capacity. This helps reduce the costs of persistent data storage for large-scale containers.

Use case 2. Multi-Pod/container data sharing (static mounting of CFS recommended)

CFS provides shared access to a directory space from multiple Pods or containers over NFS or private protocols for efficient data sharing. Compared with other schemes, this provides higher bandwidth and IOPS capabilities.

This document describes how to deploy a container workload in the Tencent Cloud console. For more information on how to write a YAML file, refer to the YAML file automatically generated after a StorageClass is created in the console.

> Note：
> Make sure that the CSI component in the TKE cluster is on v1.0.4 or later; otherwise, update it in the TKE console. The updating operation does not affect the normal use of the container.

Directions

Dynamically mounting CFS

> Note：

This mounting option is recommended for persistent storage of Pod/container data.

1. Create a StorageClass as instructed in Managing CFS Templates by Using a StorageClass.

Key configuration items are as described below:

Configuration Item	Description
Instance creation mode	Select **Shared instance**.
Availability zone	We recommend you select the same AZ as that of the container host.
Storage type	Select **Standard**. or **High-performance**. as needed.
Protocol version	Unless in scenarios involving concurrent modifications, we recommend you use the NFS v3 protocol for a higher performance.
Reclaim policy	Select **Delete**. or **Retain**. as needed. To avoid mistaken data deletion, we recommend you select **Retain**..

2. Create a PVC as instructed in Managing CFS by Using PVs and PVCs.

Key configuration items are as described below:

Configuration Item	Description
Namespace	Select a namespace as needed.
StorageClass	Select the just created StorgeClass.
PersistentVolume	You don't need to specify a PV for dynamic creation. **Note**: For a StorageClass based on a shared CFS instance, if you don't specify a PV when creating a PVC, the CSI plugin will automatically create a pay-as-you-go CFS instance when creating a PVC. This instance will be deleted when the PVC is deleted. Therefore, process PVCs created in this way with caution.

3. Create a Deployment as instructed in Deployment Management.

Key configuration items are as described below:

Configuration Item	Description
Volume	Name the volume as needed and select the just created PVC.
Mount point	Select the volume and specify the path for mounting to the container. When you dynamically create a shared CFS instance, you need to specify an environment variable, and the CSI plugin will create a directory in the CFS instance corresponding to the selected PVC based on the value of the configured environment variable for the container to mount.

After completing the configuration, click **Create Workload**, and the system will create a container based on this configuration and mount CFS.

Statically mounting CFS

> Note：
> This mounting option is recommended for multi-Pod/container data sharing.

1. Create a StorageClass as instructed in Managing CFS Templates by Using a StorageClass.

Key configuration items are as described below:

Configuration Item	Description
Instance creation mode	Select **.Shared instance**..
Availability zone	We recommend you select the same AZ as that of the container host.
Storage type	Select **.Standard**. or **.High-performance**.as needed.
Protocol version	Unless in scenarios involving concurrent modifications, we recommend you use the NFS v3 protocol for a higher performance.
Reclaim policy	Select **.Delete**. or **.Retain**. as needed. To avoid mistaken data deletion, we recommend you select **.Retain**..

2. Create a PV as instructed in Managing CFS by Using PVs and PVCs.

Key configuration items are as described below:

Configuration Item	Description
Creation method	Select **.Manual**.; that is, specify a CFS instance for PV configuration.
StorgeClass	Select the just created StorgeClass.
Select CFS	Select a specified CFS instance. **Note**: During dynamic creation, make sure that you already have a CFS instance in the same VPC as the container.
CFS sub-directory	CFS allows you to mount sub-directories. You can select different subdirectories and bind them to one or multiple PVs as needed to implement different degrees of data sharing.

3. Create a PVC as instructed in Managing CFS by Using PVs and PVCs.

Key configuration items are as described below:

Configuration Item	Description
Namespace	Select a namespace as needed.
StorageClass	Select the just created StorgeClass.
PersistentVolume	Select **.Specify**. and select the just created PV.

4. Create a Deployment as instructed in Deployment Management.

Key configuration items are as described below:

Configuration Item	Description
Volume	Name the volume as needed and select the just created PVC.
Mount point	Select the volume and specify the path for mounting to the container. To specify a sub-path of the file system for mounting, make sure that the directory exists, which does not need to start with `/`. If the directory does not exist, and you want the container to automatically create one, you can specify an environment variable, and the CSI plugin will automatically create the directory in the root path of the file system with the name of the variable value and provide it to the container for mounting.

After completing the configuration, click **Create Workload**, and the system will create a container based on this configuration and mount CFS.

Using CFS on SCF

Last updated：2022-10-12 15:59:56

Overview

Tencent Cloud Serverless Cloud Function (SCF) is a serverless execution environment that enables you to build and run applications without having to purchase and manage servers. Simply code in a supported language and set the execution conditions, and your code can be run on the Tencent Cloud infrastructure elastically and securely. SCF is an ideal computing platform for use cases such as real-time file processing and data processing.

SCF is a service-level computing resource that features fast iteration and super-fast deployment. As a result, it requires the separation of storage and computing, which makes CFS, a high-performance shared storage service, the best storage solution for SCF. With only a few easy steps, your function can easily access files stored in CFS. The benefits of using CFS on SCF are as follows:

- The execution space of functions is unlimited.
- Multiple functions can share the same file system to share files.

Directions

Associating an authorization policy

> Note：
> To use the CFS service, SCF needs permission to operate on your CFS resources.

Follow the steps below to grant the permission to your account:

1. Associate the `SCF_QcsRole` role with the `QcloudCFSReadOnlyAccess` policy as instructed in Modifying Role. The result of a successful association is shown below:
   If you don't perform this operation for your currently used account, problems such as the failure to save functions and the unavailability of CFS features may occur.

2. If the currently used account is a sub-account, request the root account to associate the sub-account with the `QcloudCFSReadOnlyAccess` policy as instructed in Setting Sub-user Permissions. The result of a successful association is shown below:

If you don't perform this operation for your currently used sub-account, problems such as the unavailability of CFS features may occur.

Creating a VPC

Build a VPC as instructed in Building Up an IPv4 VPC.

Creating CFS resources

Create a CFS file system as instructed in Creating File Systems and Mount Targets.

> Note：
> Currently, SCF allows only CFS file systems with network type being VPC to be added as mount targets. When creating a CFS file system, select the same VPC as that of the target function to enable communication.

Mounting and using a CFS file system

1. Log in to the SCF console and select **Functions** on the left sidebar.
2. On the **Function Service** page, select the name of the function to be configured.
3. On the **Function configuration** tab of the **Function management** page, click **Edit** in the top-right corner.
4. Check **Enable** for **VPC** and select the VPC where your CFS file system resides.
5. Check **Enable** for **File system** and enter the following information to mount the file system.

- **User ID** and **User group ID**: IDs of the user and user group in CFS file system. SCF uses "10000" for both the user ID and user group ID by default to manipulate your CFS file system. Set the file owner and corresponding group permission as needed and ensure that your CFS file system has the required permission. For more information, see Managing Permissions.
- **Remote directory**: The remote directory in the CFS file system to be accessed by the function, which consists of a file system and a remote directory.
- **Local directory**: Mount target of the local file system. You can use a subdirectory in the `/mnt/` directory to mount the CFS file system.
- **File System ID**: Select the file system to be mounted in the drop-down list.
- **Mount point ID**: Select the ID of the mount target corresponding to the file system in the drop-down list.

6. Click **Save** at the bottom of the page.

You can run the following function code to start using the CFS file system.

```
'use strict';
var fs = require('fs');
exports.main_handler = async (event, context) => {
await fs.promises.writeFile('/mnt/myfolder/filel.txt', JSON.stringify(event));
return event;
};
```

Performance test for using a CFS file system on SCF

You can use this demo to test how well CFS performs on SCF.

Using CFS Turbo in TKE

Last updated：2022-11-18 12:29:24

Overview

This document describes how to integrate CFS Turbo with a Tencent Cloud Kubernetes Engine (TKE) cluster.

Prerequisites

- The operating system of the TKE host node is compatible with the Turbo series.
- You have installed a Turbo-based client on all TKE nodes. You are advised to use pshell to operate in batches. For the compatible operating systems and how to install the clients, see Using CFS Turbo on Linux Clients.

Directions

Download and configure kubectl

1. Download kubectl by referring to kubectl Documentation.
2. Log in to the TKE console and select **Cluster** on the left sidebar.
3. On the **Cluster Management** page, click the ID of the target cluster to go to the cluster details page.
4. On the cluster details page, click **Basic Information**.
5. In **Cluster APIServer Information**, click **Download** to download the `kubeconfig` file to the default environment variable address and name it `config`, that is, `/usr/local/bin/config`.
6. In **Cluster APIServer Information**, set **Private Network Access** to  and click  to copy the command for host configuration.
7. Switch to the access machine to run the command copied from Step 6.

8. Run the following commands to configure the environment variable:

```
vi /etc/profile
KUBECONFIG=/usr/local/bin/config
PATH=$PATH
export KUBECONFIG
export PATH
```

9. Run the following command to verify whether kubectl has been installed successfully:

```
kubectl get node
```

If the following message is displayed, the installation is completed:

```
[root@VM-0-5-centos ~]# kubectl get node
NAME          STATUS   ROLES    AGE   VERSION
[redacted]    Ready    <none>   15d   v1.18.4-tke.9
```

Creating a Pod for mounting Turbo through YAML files

1. See TKE Turbo Plugin Guide and download the script.

2. Go to the `kubernetes-csi-tencentcloud/deploy/cfsturbo/kubernetes/` directory and upload `csi-node-rbac.yaml`, `csi-node.yaml`, and `csidriver-new.yaml` files to the kubectl management node.

3. Go to the `kubernetes-csi-tencentcloud/deploy/cfsturbo/examples/` directory and download the `pv.yaml`, `pvc.yaml`, and `pod.yaml` sample files.

4. Modify the `pv.yaml`, `pvc.yaml`, and `pod.yaml` files based on the PV, PVC, and Pod attributes, such as name and image address.

Below are the sample YAML files:

```
apiVersion: v1
kind: PersistentVolume
metadata:
name: csi-cfsturbo-pv
```

```
spec:
accessModes:
- ReadWriteMany
capacity:
storage: 10Gi
csi:
driver: com.tencent.cloud.csi.cfsturbo
# volumeHandle in PV must be unique, use pv name is better
volumeHandle: csi-cfsturbo-pv
volumeAttributes:
# cfs turbo proto
proto: lustre
# cfs turbo rootdir
rootdir: /cfs
# cfs turbo fsid (not cfs id)
fsid: d3dcc487
# cfs turbo server ip
host: 10.0.1.16
# cfs turbo subPath
path: /
storageClassName: ""
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
name: csi-cfsturbo-pvc
spec:
accessModes:
- ReadWriteMany
resources:
requests:
storage: 10Gi
# You can specify the pv name manually or just let kubernetes to bind the pv and
pvc.
volumeName: csi-cfsturbo-pv
# cfsturbo only supports static provisioning, the StorageClass name should be emp
ty.
storageClassName: ""
---
apiVersion: apps/v1
kind: Deployment
metadata:
labels:
k8s-app: csi-cfsturbo-pod
name: csi-cfsturbo-pod
spec:
replicas: 1
```

```
selector:
matchLabels:
k8s-app: csi-cfsturbo-pod
template:
metadata:
labels:
k8s-app: csi-cfsturbo-pod
spec:
containers:
- image: nginx
name: csi-cfsturbo-pod
volumeMounts:
- mountPath: /csi-cfsturbo
name: csi-cfsturbo
volumes:
- name: csi-cfsturbo
persistentVolumeClaim:
# Replaced by your pvc name.
claimName: csi-cfsturbo-pvc
```

Below is the sample command for mounting:

```
sudo mount.lustre -o sync,user_xattr 10.0.1.16@tcp0:/d3dcc487/cfs /path/to/mount
```

Below are key parameters:

- proto:lustre: Keep this parameter unchanged.
- roodir:/cfs: Keep this parameter unchanged.
- fsid:d3dcc487: Here, `fsid` is not the `CFSID`, and you need to enter the information in the mounting path.
- host:10.0.1.16: IP of the mount point.
- path: You can adjust it based on the target subpath. To directly mount the root directory, enter "/".

5. Run the following commands in sequence in the directory where the script is uploaded:

- Configure RBAC:

```
kubectl apply -f csi-node-rbac.yaml
```

- Configure the node CSI plugin:

```
kubectl apply -f csidriver-new.yaml
```

```
kubectl apply -f csi-node.yaml
```

- Create PV, PVC, and pod:

```
kubectl create -f pv.yaml
kubectl create -f pvc.yaml
kubectl create -f pod.yaml
```

6. Run the following command to view the pod status:

```
kubectl get pod -n default -o wide
```

If the message below is displayed, the pod is created successfully:

```
NAME     READY   STATUS    RESTARTS   AGE   IP          NODE          NOMINATED NODE   READINESS GATES
nginx2   1/1     Running   0          36s   10.0.0.8    192.168.0.2   <none>           <none>
```

If the value of `STATUS` is `ContainerCreating`, the creation failed. You can find the events in the TKE console to troubleshoot.

Using CFS Turbo on EKS

Last updated：2022-02-16 12:36:13

Overview

You can mount a CFS Turbo storage for an EKS cluster. The add-on is used to mount a Tencent Cloud CFS Turbo file system to a workload based on a proprietary protocol. Currently, only static configuration is supported. For more information about CFS storage types, see Storage Types and Performance.

Prerequisites

An EKS cluster of v1.14 or later has been created.

Directions

Creating a file system

Create a CFS Turbo file system. For details, see Creating File Systems and Mount Targets.

> Note：
> After the file system is created, you need to associate the cluster network (vpc-xx) with the CCN instance of the file system. You can check it in the information about the file system mount target.

Deploying a Node Plugin

Step 1. Create a csidriver.yaml file

Here is an example of a csidriver.yaml file:

```
apiVersion: storage.k8s.io/v1beta1
kind: CSIDriver
metadata:
name: com.tencent.cloud.csi.cfsturbo
spec:
attachRequired: false
podInfoOnMount: false
```

Step 2. Create a CSI driver

Run the following command to create a CSI driver:

```
kubectl apply -f csidriver.yaml
```

Creating a CFS Turbo volume

Step 1. Create a CFS Turbo type PV based on the following template

```
apiVersion: v1
kind: PersistentVolume
metadata:
name: pv-cfsturbo
spec:
accessModes:
- ReadWriteMany
capacity:
storage: 10Gi
csi:
driver: com.tencent.cloud.csi.cfsturbo
volumeHandle: pv-cfsturbo
volumeAttributes:
host: *.*.*.*
fsid: ********
storageClassName: ""
```

Parameter description:

- **metadata.name**: the name of the created PV.
- **spec.csi.volumeHandle**: it must be consistent with the PV name.
- **spec.csi.volumeAttributes.host**: the IP address of the file system. You can check it in the information about the file system mount target.
- **spec.csi.volumeAttributes.fsid**: fsid of the file system (not the file system ID). You can check it in the information about the file system mount target. It is the string after "tcp0:/" and before "/cfs" in the mount command, as shown below.

Step 2. Create a PVC that is bound to the PV based on the following template

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
name: pvc-cfsturbo
spec:
```

```
storageClassName: ""
volumeName: pv-cfsturbo
accessModes:
- ReadWriteMany
resources:
requests:
storage: 10Gi
```

Parameter description:

- **metadata.name**: the name of the created PVC.
- **spec.volumeName**: it must be consistent with the name of the PV created in Step 1.

Using a CFS Turbo volume

Create a Pod that mounts the PVC based on the following template.

```
apiVersion: v1
kind: Pod
metadata:
name: nginx
spec:
containers:
- image: ccr.ccs.tencentyun.com/qcloud/nginx:1.9
imagePullPolicy: Always
name: nginx
ports:
- containerPort: 80
protocol: TCP
volumeMounts:
- mountPath: /var/www
name: data
volumes:
- name: data
persistentVolumeClaim:
claimName: pvc-cfsturbo
```

Network Selection for Turbo CFS

Last updated：2022-11-18 12:32:44

CFS Turbo is Tencent Cloud's high-performance parallel file system. Compared with traditional general CFS, it supports CCN and VPC networks on the client and server, both of which have their respective pros and cons. This document describes the two network types to help you select a more suitable one for your business. If you use CCN, we recommend you create Turbo file systems based on CCN; otherwise, you can select VPC for easier use, if applicable.

CCN

Overview

Specified IP ranges are assigned to Turbo file systems, and CCN capabilities are leveraged to connect the VPC and storage server network, implementing the interaction between computing instances and storage.

Pro	Con
• IP range planning is separately stored for more efficient and convenient management of security groups. • Turbo CFS has its own IP range to reserve sufficient IPs for further scale-outs without limits. • Turbo CFS can be accessed more easily across VPCs. • IPs of the existing VPC are not occupied.	CCN is required for network connection. If CCN is not used, a new component needs to be introduced, which is quite complicated.

Best practices

An existing CCN instance or a newly created one is used to assign 9/11/30 Tencent server IP ranges to Turbo CFS, without occupying business IPs.

VPC

Overview

IPs are mapped to the existing VPC on the storage server for mount and access.

Pro	Con

Pro	Con
• It is the easiest to use and is similar to general CFS. • No new components need to be introduced in this simple solution.	• During large scale-outs, subnet IPs may be insufficient. • VPC IPs will be occupied, and the storage is bound to the VPC, which hinders network isolation.