Cloud File Storage

Best Practice

Contents

Best Practice
Selecting Kernels for NFS Clients

Last updated：2024-12-12 15:35:26

The NFS client is a kernel-based client. Due to bugs in some kernel versions, the NFS service may not function properly. To ensure a better user experience, we recommend using our suggested kernel versions.

Identified Client Issues

Kernel network stack defects causing unresponsive file system (Priority: High)

When the system's kernel version is 2.6.32-696~2.6.32-696.10.1 (including 2.6.32-696, but excluding 2.6.32-696.10.1), the NFS server is busy and the kernel requests retransmission, which may trigger kernel network stack defects, resulting in unresponsive operations. If the operation is unresponsive, please restart the CVM instance. For more information, please refer to RHEL6.9:NFSv4 TCP transport stuck in FIN_WAIT_2 forever.

Kernel defects causing unresponsive file system (Priority: High)

- When the system's kernel version is one of the following, a failover in the NFS server may cause deadlock situations in the NFS client's open, read, and write operations, leading to a continuous unresponsive state of the file system.
  - Redhat 6、CentOS 6 2.6.32-696.3.1.el6.
  - All kernel versions prior to Redhat 7, CentOS 7 3.10.0-229.11.1.el7.
- Ubuntu 15.10 Linux 4.2.0-18-generic. If the operation is unresponsive, please restart the CVM instance. For more information, please refer to RHEL7:NFSv4 client loops with WRITE/NFS4ERR_STALE_STATEID - if NFS server restarts multiple times within the grace period.
- When the system's kernel version is one of the following versions, network partitioning or jitter may cause connection reconnection. The NFS client may continuously be unresponsive due to incorrect error code handling. The symptom is an unresponsive file system and the repeated printing of a bad sequence-id error in the system message.
  - All kernel versions prior to Redhat 6, CentOS 6 2.6.32-696.16.1.el6.
  - All kernel versions prior to Red Hat 7, CentOS 7 3.10.0-693.el7. If the operation is unresponsive, please restart the CVM instance. For more information, please refer to RHEL6/RHEL7:NFS4 client receiving NFS4ERR_BAD_SEQID drops nfs4 stateowner resulting in infinite loop of READ/WRITE+NFS4ERR_BAD_STATEID.

- When the operating system kernel version is all kernels of CentOS and RedHat 5.11.x, executing the ls command, commands containing wildcard * or ?, and other operations that require directory traversal, will all cause lag or unresponsiveness due to kernel defects. Please upgrade your kernel version to avoid this issue.

Unsupported chown command and system call (Priority: Low)

When the system's kernel version is 2.6.32, it does not support the execution of the chown command and system calls by the NFS client.

Inability to terminate ls operation (Priority: Low)

- When the system's kernel version is 2.6.32-696.1.1.el6 or earlier, executing the 'ls' command while adding or deleting files and subdirectories will cause the 'ls' operation to never terminate. Please upgrade the kernel version to avoid this issue.
- When the system's kernel version is 4.18.0-305.12.1, directory traversal operations such as ls may not terminate. Please upgrade the kernel to 4.18.0-305.19.1 to fix this issue.

Recommended Images for NFS File System

Linux System Image

Operating system	OS Version
CentOS	• CentOS 6.9 64-bit: 2.6.32-696.16.1.el6.x86_64 and above • CentOS 6.10 64-bit: 2.6.32-754.17.1.el6.x86_64 and above • CentOS 7.2 64-bit: 3.10.0-514.26.2.el7.x86_64 and above • CentOS 7.3 64-bit: 3.10.0-514.26.2.el7.x86_64 and above • CentOS 7.4 64-bit: 3.10.0-693.2.2.el7.x86_64 and above • CentOS 7.5 64-bit: 3.10.0-862.14.4.el7.x86_64 and above • CentOS 7.6 64-bit: 3.10.0-957.21.3.el7.x86_64 or above • CentOS 7.7 64-bit: 3.10.0-1062.18.1.el7.x86_64 and above • CentOS 8.x 64-bit: 4.18.0-147.5.1.el8_1.x86_64 and above

Tencent OS Linux	• TencentOS Server 2.2(Tkernel 3) • TencentOS Server 2.4 (Tkernel 4) • TencentOS Server 2.6(Final) • TencentOS Server 3.1(Tkernel 4)
Debian	• Debian 9.6 64-bit: 4.9.0-8-amd64 and above • Debian 9.8 64-bit: 4.9.0-8-amd64 and above • Debian 9.10 64-bit: 4.9.0-9-amd64 and above
Ubuntu	• Ubuntu 14.04 64-bit: 4.4.0-93-generic and above • Ubuntu 16.04 64-bit: 4.4.0-151-generic and above • Ubuntu 18.04 64-bit: 4.15.0-52-generic and above • Ubuntu 20.04 64-bit: 5.4.0-31-generic and above
OpenSuse	• OpenSuse 42.3 64-bit: 4.4.90-28-default and above
Suse	• Enterprise Server 12 SP2 64-bit: 4.4.74-92.35-default and above • Enterprise Server 12 SP4 64-bit: 4.12.14-95.16-default and above
CoreOS	• CoreOS 1745.7.0 64-bit: 4.19.56-coreos-r1 and above • CoreOS 2023.4.0 64-bit: 4.19.56-coreos-r1 and above

Windows System Image

Operating system	OS Version
Windows Server 2012	• Windows Server 2012 R2 Datacenter 64-bit Chinese • Windows Server 2012 R2 Datacenter 64-bit English

Windows Server 2016	• Windows Server 2016 DataCenter 64-bit Chinese • Windows Server 2016 Datacenter 64-bit English
Windows Server 2019	• Windows Server 2019 Datacenter 64-bit Chinese • Windows Server 2019 IDC 64-bit English

Managing Turbo CFS Directories

Last updated：2023-08-29 10:40:47

Use Cases

This practice aims to achieve higher performance and lower latency of a Turbo file system by better allocating directories and files.

Background

- When clients access a file system, operations are performed based on the virtual file system (VFS) layer. The VFS layer serially grants and releases locks when processing multiple processes reading and writing the same file. If a client's I/O is under an extremely large directory at a certain level, the serial lock operations of VFS may cause increased I/O latency.
- As a strongly consistent file system, a Turbo file system maintains consistency for any client to read and write at any time. However, this also means that clients' I/O operations will involve the granting or release of backend distributed locks. The Turbo backend can handle lock requests concurrently based on distributed metadata services. However, to ensure good performance in terms of latency, we recommend you manage the number of directories and files according to the suggestions in this document.

Suggestions

- If your business mainly involves reading and there are no frequent delete/update/create operations, we recommend you control the number of subdirectories and files in a single directory to be between 10 and 1 million.
- If your business involves frequent delete/update/create operations, we recommend you control the number of subdirectories and files in a single directory to be between 10 and 10,000.

Common directory structures

- Sort by hash code: A 64-character hash code is used, where the first two characters form a first-level subdirectory, the next two characters form a second-level subdirectory, and files are stored under the second-level subdirectory. When the statistics feature of hash functions works properly, files can be evenly distributed among these 65,536 directories. When the file system has 600 million files, each directory has an average of 10,000 files. When the file system has 1.2 billion files, each directory has an average of 20,000 files.
- Sort by time: The year, month, and day form the first-level subdirectory, the hour forms the second-level subdirectory, and the minute forms the third-level subdirectory (which can

be omitted).

Terminating Compute Instances

Last updated：2023-08-29 10:41:09

Use Cases

This practice is primarily applicable when using the Turbo file system and needing to dynamically adjust computing instances. It provides an optimal method for instance termination to avoid lock conflicts caused by improper operations.
Typically, when users terminate a CVM machine, they directly call the TerminateInstances interface for termination. This interface effectively forces the server to shut down (similar to cutting off the power) and return. The advantage of this method is its speed, but for a strongly consistent distributed file system like CFS Turbo, a forced disconnection on the client side can cause abnormal lock recall on the server side, leading to IO Hang. To avoid affecting the normal use of customer business, it is recommended to use the termination method suggested in this document, which can achieve a no-impact effect on the business.

Instructions

Shutting down CVM instances

Invoke the StopInstances interface to shut down, choosing SOFT_FIRST for Stoptype. This method prioritizes a normal shutdown first, and if the instance does not shut down normally within 5 minutes, a forced shutdown is executed. This approach ensures the normal use of Turbo while considering the timeliness of the shutdown.

Note

1. Do not set `StopType` to `HARD` (forced shutdown).
2. When you perform a shutdown using the console, do not select forced shutdown. Select `SOFT_FIRST` .
3. A soft shutdown terminates processes in an orderly manner and performs a sync operation, minimizing the impact of a forced shutdown on the distributed lock in a Turbo file system.

Terminating Instances

Invoke the DescribeInstancesStatus to query the instance status. When the status is STOPPED, execute the TerminateInstances interface for termination.

Using CFS on TKE

Last updated: 2023-08-29 10:41:17

Scenario

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.

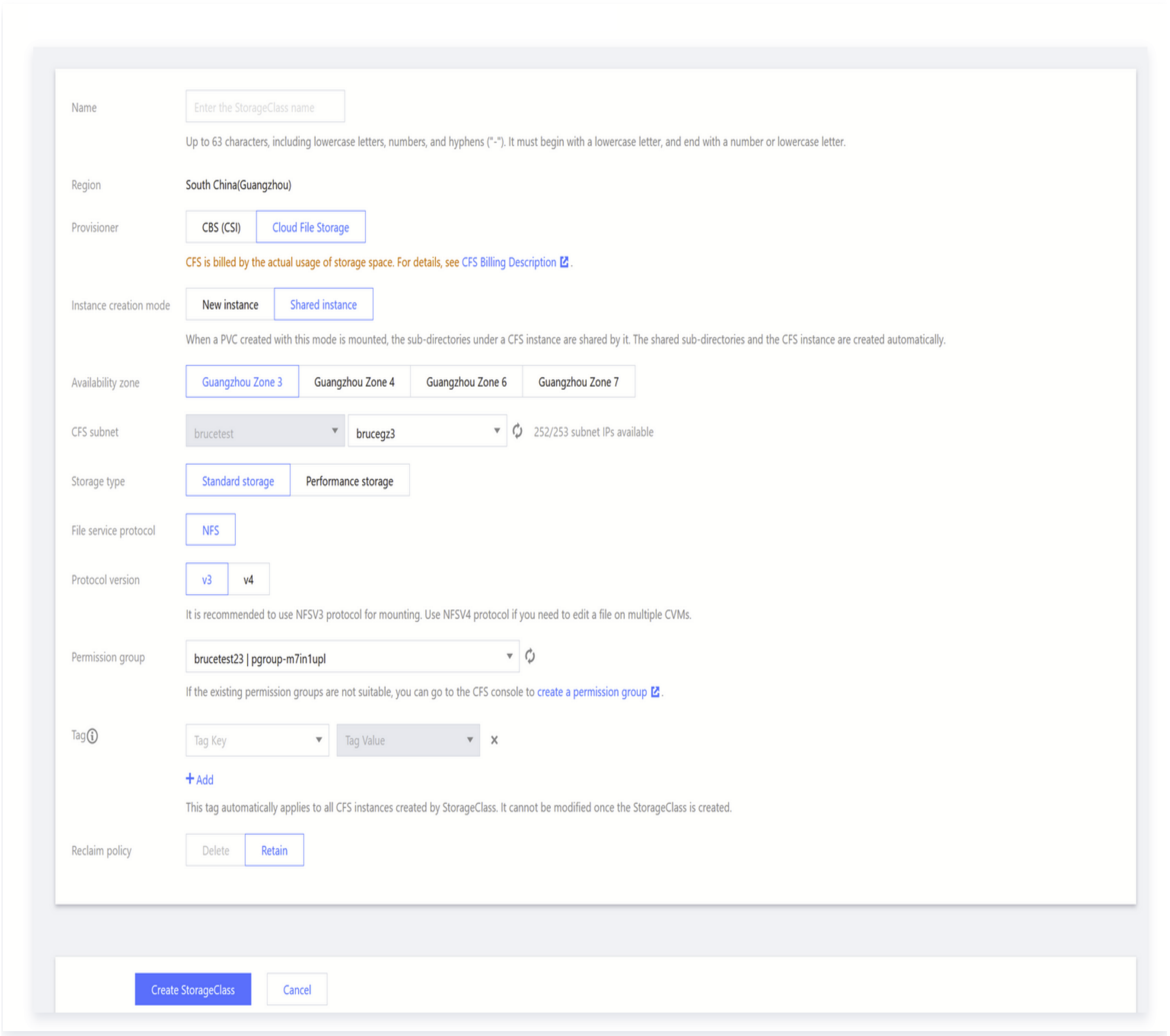
> **Note**
> Make sure that the CSI component in the TKE cluster is on v1.0.4 or later; otherwise, update it in the TKE console. The updating operation does not affect the normal use of the container.

Instructions

Dynamically mounting CFS

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

1. Create a StorageClass. For detailed operations, please refer to the Create StorageClass document.



Name: Enter the StorageClass name
Up to 63 characters, including lowercase letters, numbers, and hyphens ("-"). It must begin with a lowercase letter, and end with a number or lowercase letter.

Region: South China(Guangzhou)

Provisioner: CBS (CSI) [✓ Cloud File Storage]
CFS is billed by the actual usage of storage space. For details, see CFS Billing Description.

Instance creation mode: New instance [✓ Shared instance]
When a PVC created with this mode is mounted, the sub-directories under a CFS instance are shared by it. The shared sub-directories and the CFS instance are created automatically.

Availability zone: [✓ Guangzhou Zone 3] Guangzhou Zone 4 Guangzhou Zone 6 Guangzhou Zone 7

CFS subnet: brucetest brucegz3 252/253 subnet IPs available

Storage type: [✓ Standard storage] Performance storage

File service protocol: [✓ NFS]

Protocol version: [✓ v3] v4
It is recommended to use NFSV3 protocol for mounting. Use NFSV4 protocol if you need to edit a file on multiple CVMs.

Permission group: brucetest23 | pgroup-m7in1upl
If the existing permission groups are not suitable, you can go to the CFS console to create a permission group.

Tag: Tag Key Tag Value ×
+Add
This tag automatically applies to all CFS instances created by StorageClass. It cannot be modified once the StorageClass is created.

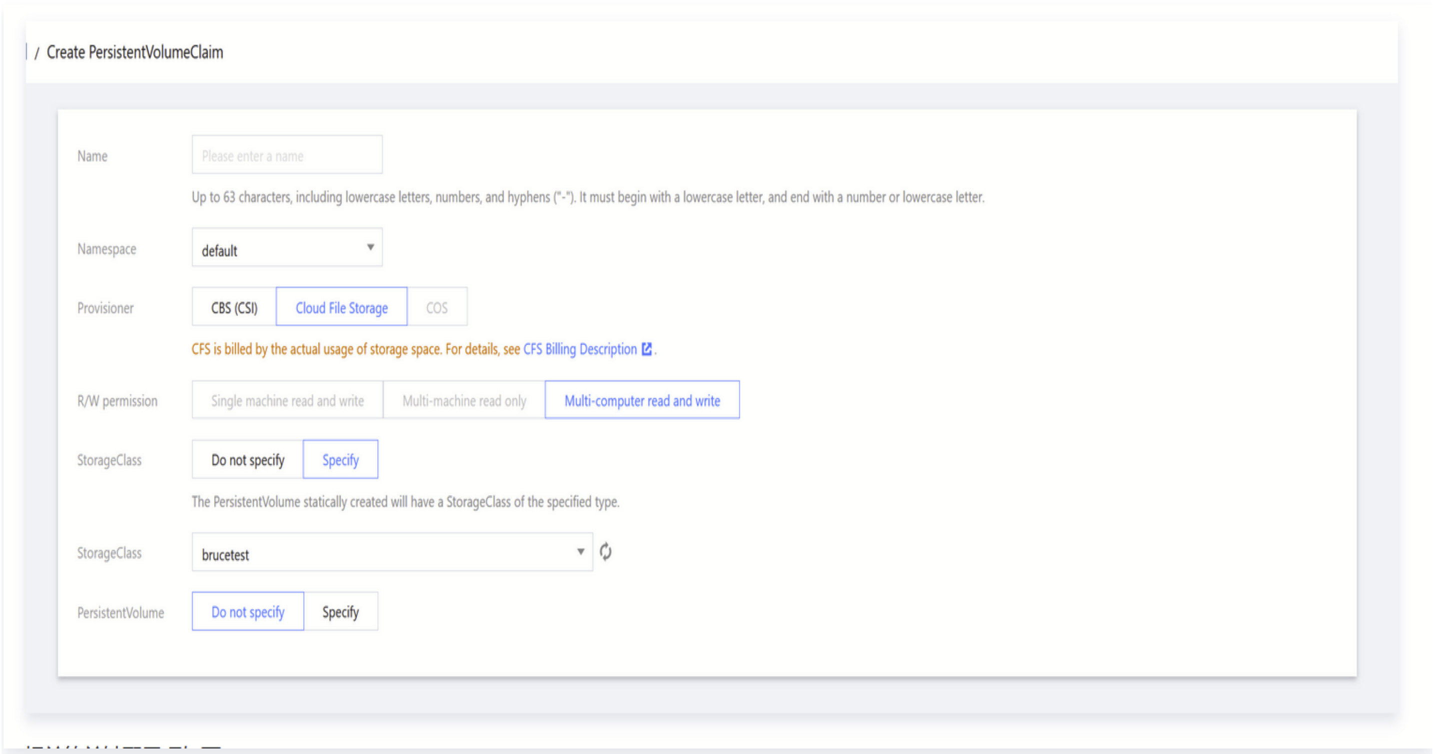
Reclaim policy: Delete [✓ Retain]

The key configuration items are as follows:

Configuration items	Description
Instance creation mode	Select Shared instance.

Availability Zones	We recommend you select the same AZ as that of the container host.
Storageclass	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 unexpected data deletion, we recommend you select Retain.

2. Create a PVC. For detailed instructions, refer to the Create PVC document.



/ Create PersistentVolumeClaim

Name: Please enter a name
Up to 63 characters, including lowercase letters, numbers, and hyphens ("-"). It must begin with a lowercase letter, and end with a number or lowercase letter.

Namespace: default

Provisioner: CBS (CSI) | Cloud File Storage | COS
CFS is billed by the actual usage of storage space. For details, see CFS Billing Description.

R/W permission: Single machine read and write | Multi-machine read only | Multi-computer read and write

StorageClass: Do not specify | Specify
The PersistentVolume statically created will have a StorageClass of the specified type.

StorageClass: brucetest

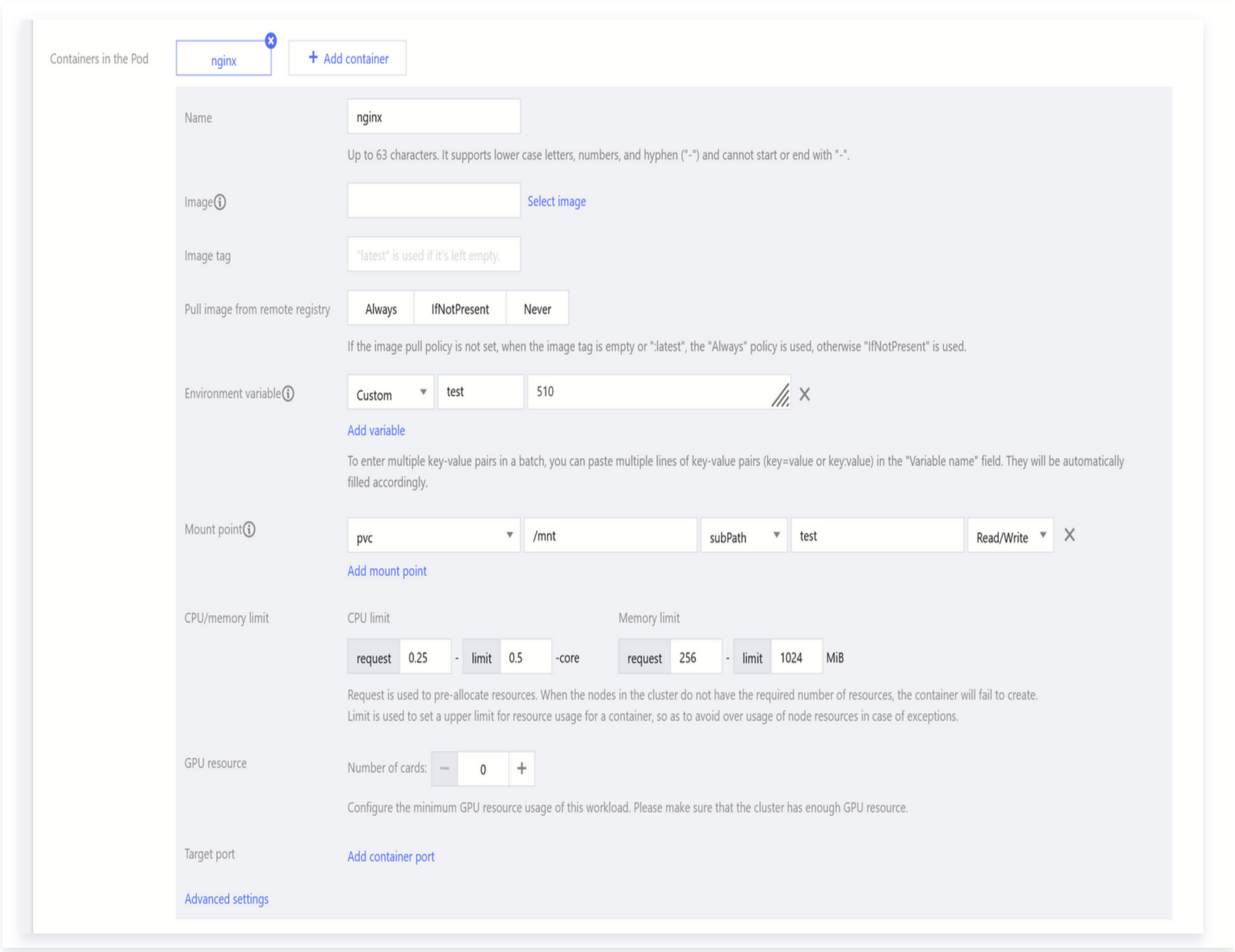
PersistentVolume: Do not specify | Specify

The key configuration items are as follows:

Configuration items	Configuration Items
Namespace	Select a namespace as needed.
Storageclass	Select the StorgeClass you just created.

Persistent Volume	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. For detailed instructions, refer to the Create Deployment document.


Containers in the Pod: nginx | Add container

Name: nginx
Up to 63 characters. It supports lower case letters, numbers, and hyphen ("-") and cannot start or end with "-".

Image: Select image

Image tag: "latest" is used if it's left empty.

Pull image from remote registry: Always | IfNotPresent | Never
If the image pull policy is not set, when the image tag is empty or ":latest", the "Always" policy is used, otherwise "IfNotPresent" is used.

Environment variable: Custom | test | 510
Add variable
To enter multiple key-value pairs in a batch, you can paste multiple lines of key-value pairs (key=value or key:value) in the "Variable name" field. They will be automatically filled accordingly.

Mount point: pvc | /mnt | subPath | test | Read/Write
Add mount point

CPU/memory limit: CPU limit: request 0.25 - limit 0.5 -core; Memory limit: request 256 - limit 1024 MiB
Request is used to pre-allocate resources. When the nodes in the cluster do not have the required number of resources, the container will fail to create.
Limit is used to set a upper limit for resource usage for a container, so as to avoid over usage of node resources in case of exceptions.

GPU resource: Number of cards: 0
Configure the minimum GPU resource usage of this workload. Please make sure that the cluster has enough GPU resource.

Target port: Add container port

Advanced settings

The key configuration items are as follows:

Configuration items	Configuration Items

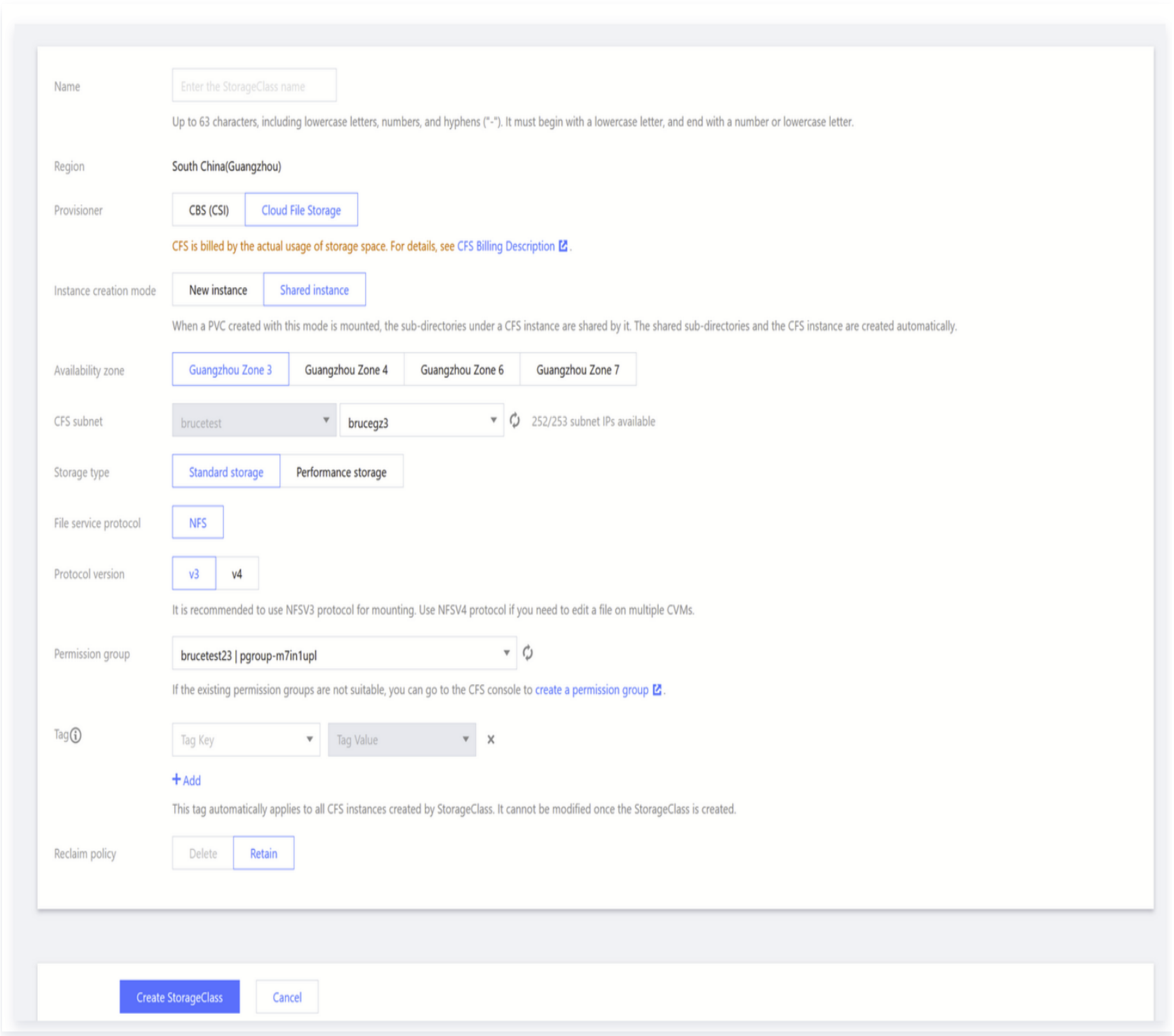
Volume	Name the volume as needed and select the PVC you just created.
Mount point	Select the volume and specify the path for mounting to the container. When you dynamically create a shared CFS instance, you need to specify an environment variable, and the CSI plugin will create a directory in the CFS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

Upon completion of the configuration, click **Create Workload** and the system will create a container based on this configuration and mount the CFS.

Statically mounting CFS

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

1. Create a StorageClass. For detailed operations, please refer to the Create StorageClass document.



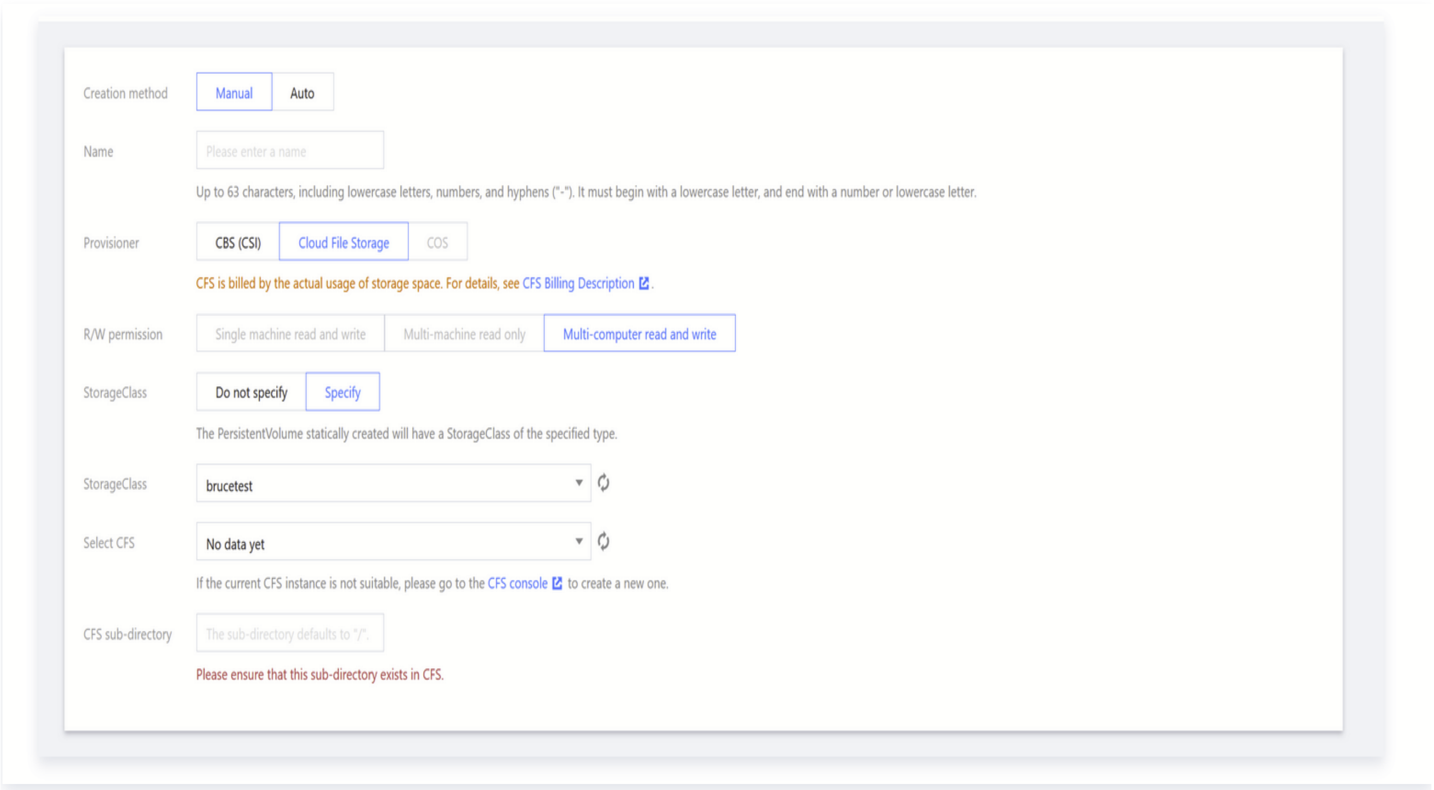
Name	Enter the StorageClass name
	Up to 63 characters, including lowercase letters, numbers, and hyphens ("-"). It must begin with a lowercase letter, and end with a number or lowercase letter.
Region	South China(Guangzhou)
Provisioner	CBS (CSI) / [✓ Cloud File Storage]
	CFS is billed by the actual usage of storage space. For details, see CFS Billing Description.
Instance creation mode	New instance / [✓ Shared instance]
	When a PVC created with this mode is mounted, the sub-directories under a CFS instance are shared by it. The shared sub-directories and the CFS instance are created automatically.
Availability zone	[✓ Guangzhou Zone 3] / Guangzhou Zone 4 / Guangzhou Zone 6 / Guangzhou Zone 7
CFS subnet	brucetest / brucegz3 — 252/253 subnet IPs available
Storage type	[✓ Standard storage] / Performance storage
File service protocol	[✓ NFS]
Protocol version	[✓ v3] / v4
	It is recommended to use NFSV3 protocol for mounting. Use NFSV4 protocol if you need to edit a file on multiple CVMs.
Permission group	brucetest23 | pgroup-m7in1upl
	If the existing permission groups are not suitable, you can go to the CFS console to create a permission group.
Tag	Tag Key / Tag Value
	+ Add
	This tag automatically applies to all CFS instances created by StorageClass. It cannot be modified once the StorageClass is created.
Reclaim policy	Delete / [✓ Retain]

The key configuration items are as follows:

Configuration items	Description
Instance creation mode	Select Shared instance.
Availability Zones	We recommend you select the same AZ as that of the container host.
Storageclass	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 unexpected data deletion, we recommend you select Retain.

2. Create a PV, for detailed operations, please refer to the Create PV statically documentation.


Creation method [✓ Manual] Auto

Name Please enter a name

Up to 63 characters, including lowercase letters, numbers, and hyphens ("-"). It must begin with a lowercase letter, and end with a number or lowercase letter.

Provisioner CBS (CSI) [✓ Cloud File Storage] COS

CFS is billed by the actual usage of storage space. For details, see CFS Billing Description .

R/W permission Single machine read and write Multi-machine read only [✓ Multi-computer read and write]

StorageClass Do not specify [✓ Specify]

The PersistentVolume statically created will have a StorageClass of the specified type.

StorageClass brucetest

Select CFS No data yet

If the current CFS instance is not suitable, please go to the CFS console to create a new one.

CFS sub-directory The sub-directory defaults to "/".

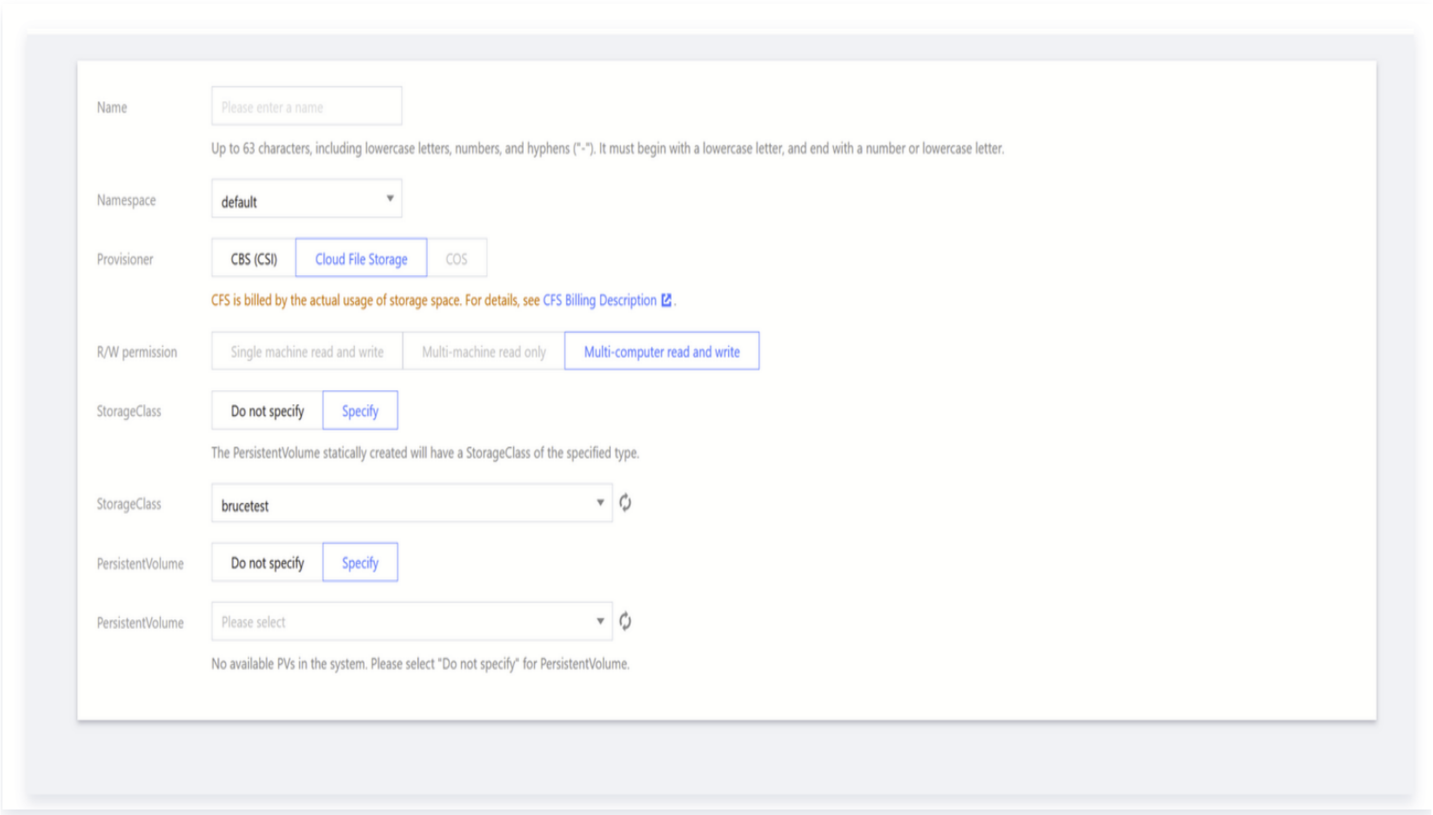
Please ensure that this sub-directory exists in CFS.

The key configuration items are as follows:

Configuration items	Configuration Items
Creation method	Creation methodSelect Manual; that is, specify a CFS instance for PV configuration.
StorgeClass	Select the StorgeClass you just created.
Select CFS	Select a specified CFS instance. Note: During static creation, make sure that you already have a CFS instance in the same VPC as the container.

CFS subdirectory	CFS allows you to mount subdirectories. 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. For detailed instructions, refer to the Create PVC document.

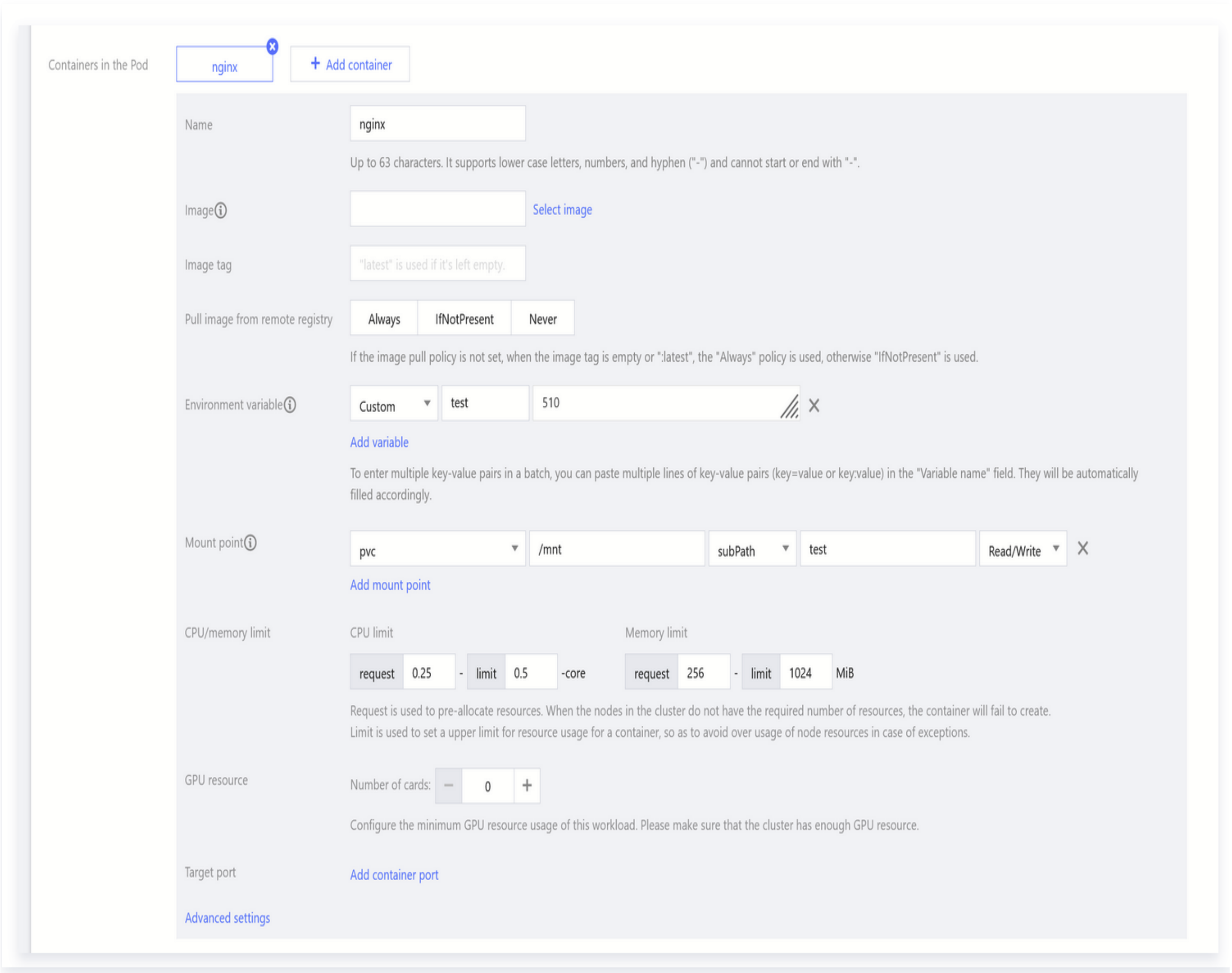


Name	Please enter a name
	Up to 63 characters, including lowercase letters, numbers, and hyphens ("-"). It must begin with a lowercase letter, and end with a number or lowercase letter.
Namespace	default
Provisioner	CBS (CSI) / [✓ Cloud File Storage] / COS
	CFS is billed by the actual usage of storage space. For details, see CFS Billing Description.
R/W permission	Single machine read and write / Multi-machine read only / [✓ Multi-computer read and write]
StorageClass	Do not specify / [✓ Specify]
	The PersistentVolume statically created will have a StorageClass of the specified type.
StorageClass	brucetest
PersistentVolume	Do not specify / [✓ Specify]
PersistentVolume	Please select
	No available PVs in the system. Please select "Do not specify" for PersistentVolume.

The key configuration items are as follows:

Configuration items	Configuration Items
Namespace	Select a namespace as needed.
Storageclass	Select the StorgeClass you just created.
PersistentVolume	Select Specify and select the PV you just created.

4. Create a Deployment. For detailed instructions, refer to the Create Deployment document.



Containers in the Pod: nginx | + Add container

Name: nginx
Up to 63 characters. It supports lower case letters, numbers, and hyphen ("-") and cannot start or end with "-".

Image: Select image

Image tag: "latest" is used if it's left empty.

Pull image from remote registry: Always | IfNotPresent | Never
If the image pull policy is not set, when the image tag is empty or ":latest", the "Always" policy is used, otherwise "IfNotPresent" is used.

Environment variable: Custom | test | 510
Add variable
To enter multiple key-value pairs in a batch, you can paste multiple lines of key-value pairs (key=value or key:value) in the "Variable name" field. They will be automatically filled accordingly.

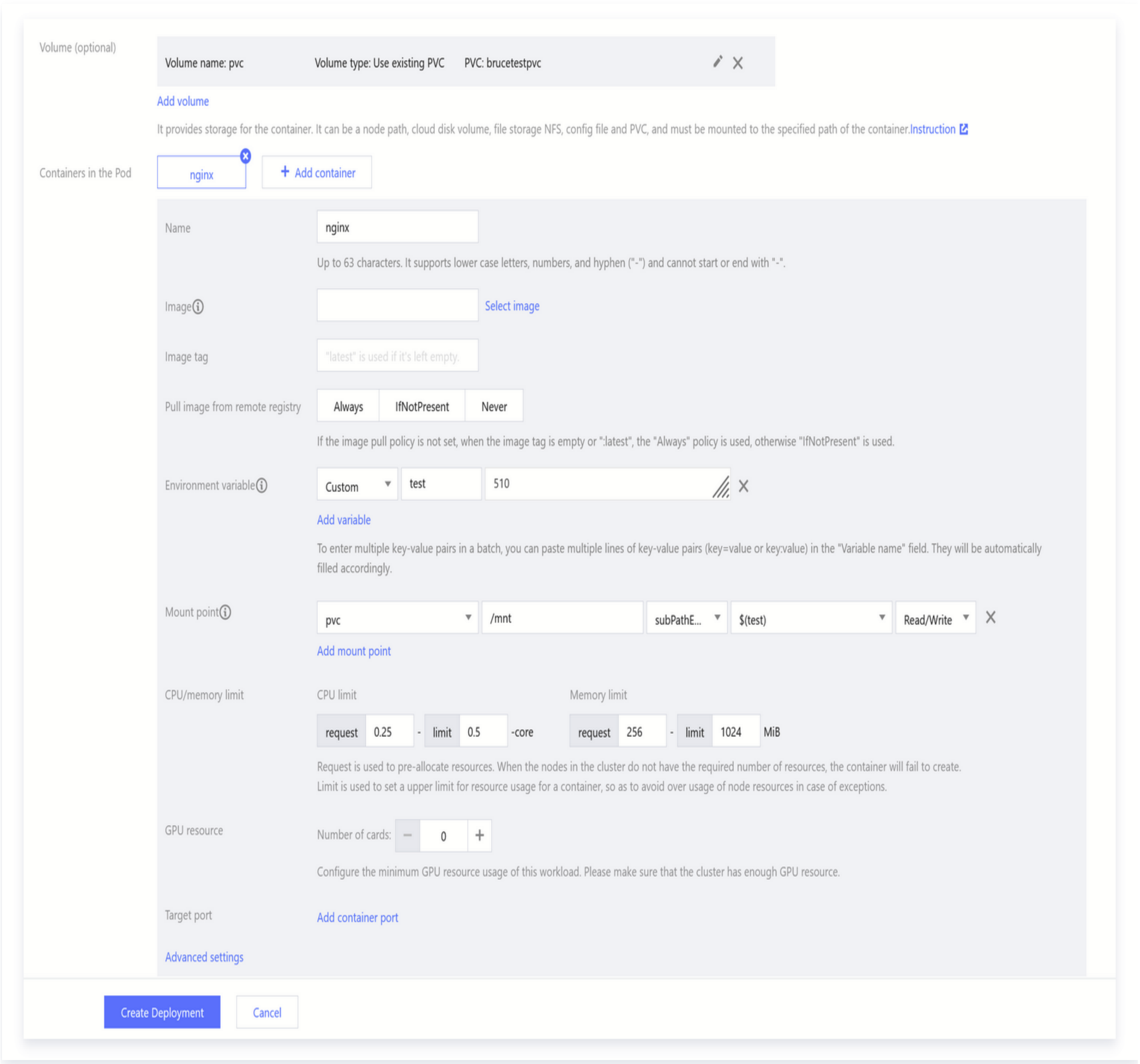
Mount point: pvc | /mnt | subPath | test | Read/Write
Add mount point

CPU/memory limit:
CPU limit: request 0.25 - limit 0.5 -core
Memory limit: request 256 - limit 1024 MiB
Request is used to pre-allocate resources. When the nodes in the cluster do not have the required number of resources, the container will fail to create.
Limit is used to set a upper limit for resource usage for a container, so as to avoid over usage of node resources in case of exceptions.

GPU resource: Number of cards: 0
Configure the minimum GPU resource usage of this workload. Please make sure that the cluster has enough GPU resource.

Target port: Add container port

Advanced settings


Volume (optional)
Volume name: pvc Volume type: Use existing PVC PVC: brucetestpvc
Add volume
It provides storage for the container. It can be a node path, cloud disk volume, file storage NFS, config file and PVC, and must be mounted to the specified path of the container. Instruction
Containers in the Pod: nginx + Add container
Name: nginx
Up to 63 characters. It supports lower case letters, numbers, and hyphen ("-") and cannot start or end with "-".
Image: Select image
Image tag: "latest" is used if it's left empty.
Pull image from remote registry: Always IfNotPresent Never
If the image pull policy is not set, when the image tag is empty or ":latest", the "Always" policy is used, otherwise "IfNotPresent" is used.
Environment variable: Custom test 510
Add variable
To enter multiple key-value pairs in a batch, you can paste multiple lines of key-value pairs (key=value or key:value) in the "Variable name" field. They will be automatically filled accordingly.
Mount point: pvc /mnt subPathE... $(test) Read/Write
Add mount point
CPU/memory limit: CPU limit request 0.25 - limit 0.5 -core; Memory limit request 256 - limit 1024 MiB
Request is used to pre-allocate resources. When the nodes in the cluster do not have the required number of resources, the container will fail to create.
Limit is used to set a upper limit for resource usage for a container, so as to avoid over usage of node resources in case of exceptions.
GPU resource: Number of cards: 0
Configure the minimum GPU resource usage of this workload. Please make sure that the cluster has enough GPU resource.
Target port: Add container port
Advanced settings
Create Deployment Cancel

The key configuration items are as follows:

Configuration items	Configuration Items
Volume	Name the volume as needed and select the PVC you just created.

Mount point	Select the volume and specify the path for mounting to the container. To achieve automatic subdirectory creation, you can use two methods. Method 1: Select subPath as the mount point and enter the name of the path you want to mount, such as test. Then, the CSI plugin will automatically create the test directory under the root path of the file system and automatically mount it to the specified directory of the container. Method 2: Add an environment variable and assign a value to it. Select subPathExpr as the mount point and select the environment variable. The CSI plugin will use the value of the environment variable as the directory name, automatically create the directory under the root path of the file system, and automatically mount it to the specified directory of the container.

Upon completion of the configuration, click **Create Workload** and the system will create a container based on this configuration and mount the CFS.

Using CFS on SCF

Last updated：2023-08-29 10:41:26

Scenario

Serverless Cloud Function (SCF) is a serverless execution environment provided by Tencent Cloud for businesses and developers. It allows you to run code without the need to purchase and manage servers. All you need to do is write your core code in a supported language and set the conditions for the code to run. This allows for flexible and secure code execution on Tencent Cloud's infrastructure. SCF is an ideal computing platform for real-time file processing and data processing scenarios.
SCF is a service-level computing resource. Its rapid iteration and swift deployment characteristics inherently require a separation of storage and computation. Cloud File Storage (CFS) provides a high-performance shared storage service, making it the optimal storage solution for SCF. With just a few simple configurations, your functions can easily access files stored in the CFS file system. The advantages of SCF using CFS are as follows:

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

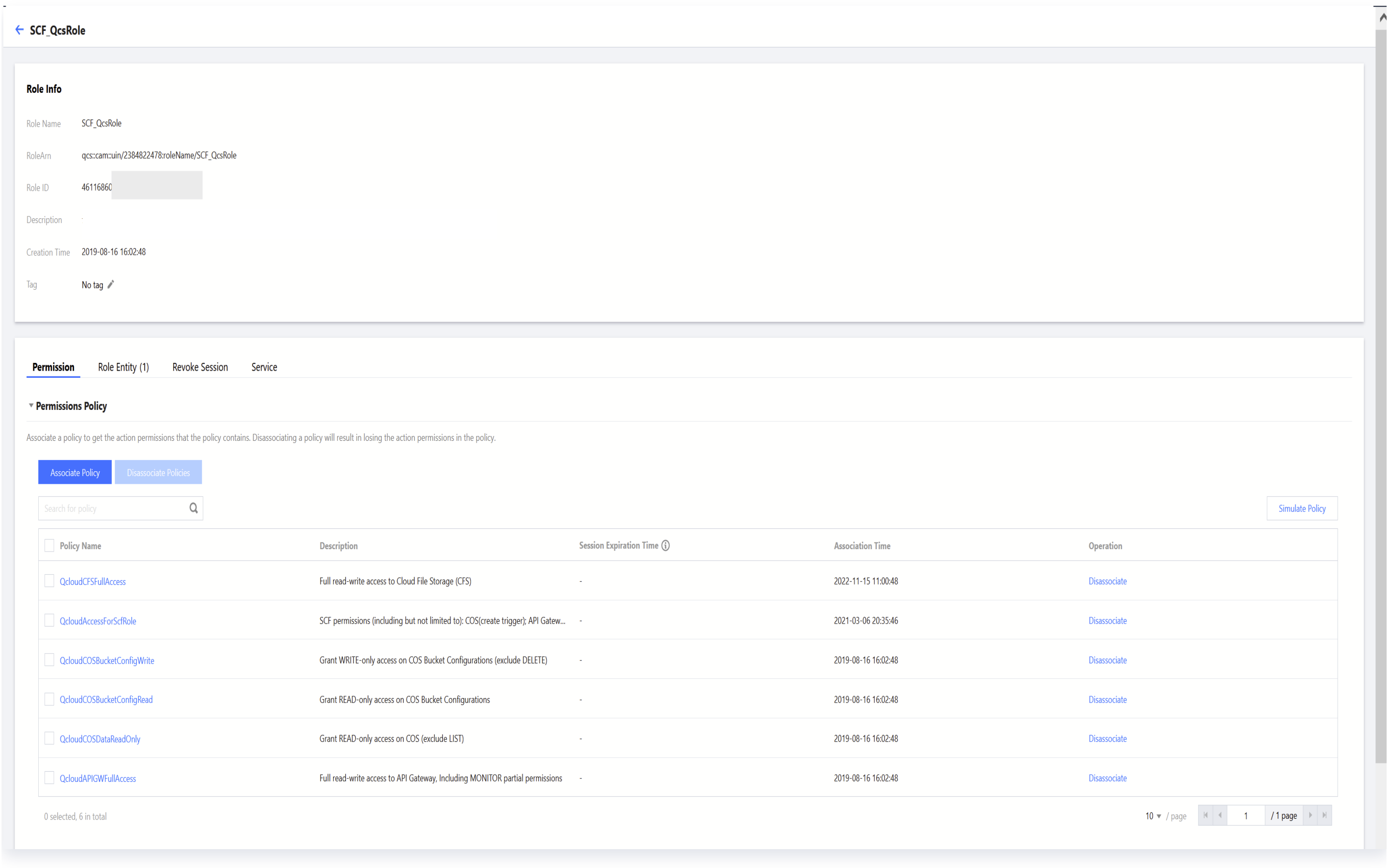
Instructions

Associating an authorization policy

Note
To use the CFS feature, your cloud function needs permission to operate your CFS resources.

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

1. Please refer to Modify Role to associate the `SCF_QcsRole` role with the `QcloudCFSReadOnlyAccess` policy. Successful association is as shown in the figure below:
   If the account you are using has not performed this operation, you may encounter issues such as the function not being able to save, and CFS related functions not being usable.



SCF_QcsRole

Role Info

Role Name	SCF_QcsRole
RoleArn	qcs::cam::uin/2384822478:roleName/SCF_QcsRole
Role ID	46116860[redacted]
Description	-
Creation Time	2019-08-16 16:02:48
Tag	No tag

Permission | Role Entity (1) | Revoke Session | Service

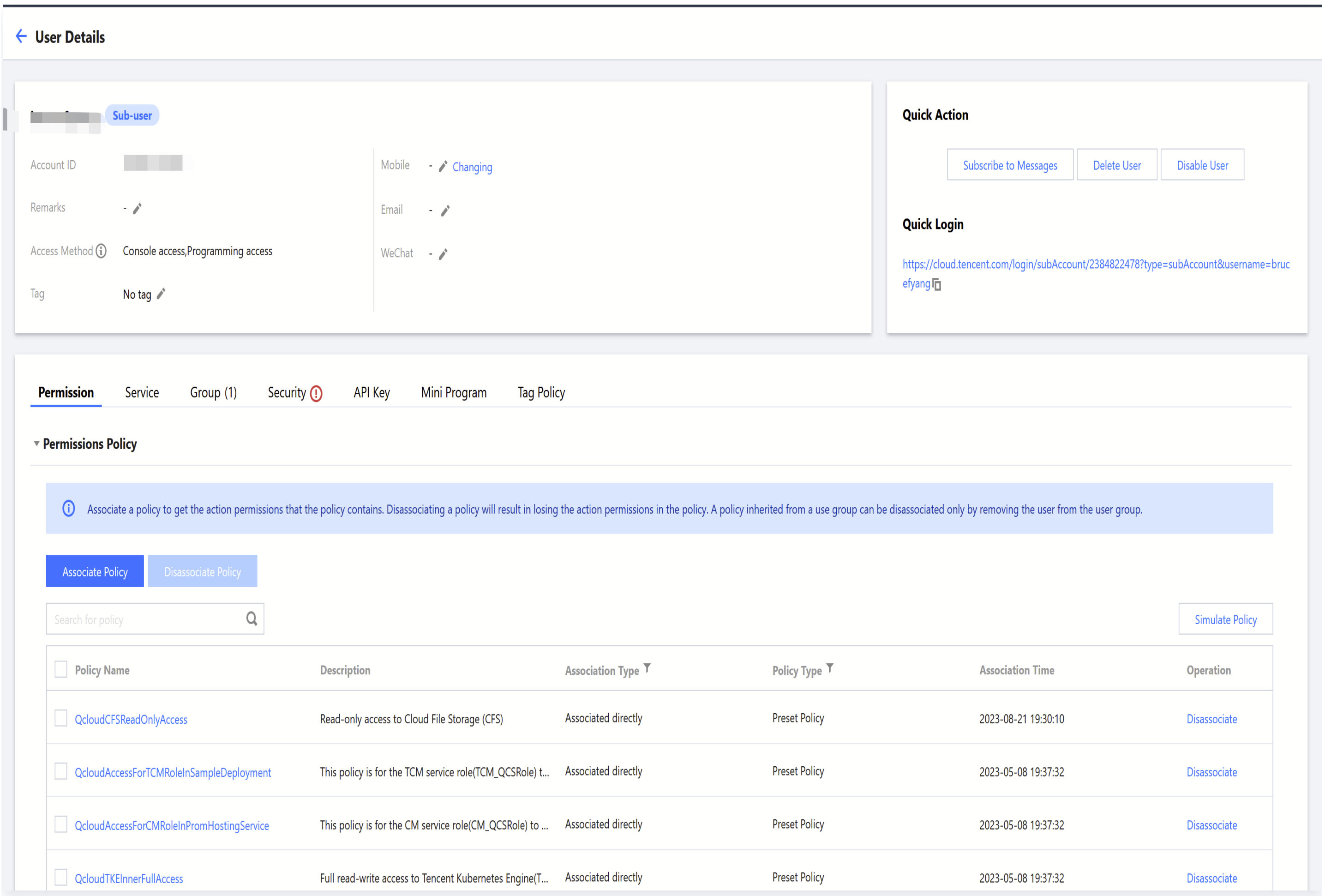
Permissions Policy

Associate a policy to get the action permissions that the policy contains. Disassociating a policy will result in losing the action permissions in the policy.

Policy Name	Description	Session Expiration Time	Association Time	Operation
QcloudCFSFullAccess	Full read-write access to Cloud File Storage (CFS)	-	2022-11-15 11:00:48	Disassociate
QcloudAccessForScfRole	SCF permissions (including but not limited to): COS(create trigger); API Gatew...	-	2021-03-06 20:35:46	Disassociate
QcloudCOSBucketConfigWrite	Grant WRITE-only access on COS Bucket Configurations (exclude DELETE)	-	2019-08-16 16:02:48	Disassociate
QcloudCOSBucketConfigRead	Grant READ-only access on COS Bucket Configurations	-	2019-08-16 16:02:48	Disassociate
QcloudCOSDataReadOnly	Grant READ-only access on COS (exclude LIST)	-	2019-08-16 16:02:48	Disassociate
QcloudAPIGWFullAccess	Full read-write access to API Gateway, Including MONITOR partial permissions	-	2019-08-16 16:02:48	Disassociate

0 selected, 6 in total

2. If you are using a sub-account, please contact the root account and refer to Sub-user Permission Settings to associate your sub-account with the `QcloudCFSReadOnlyAccess` policy. Successful association is as shown in the figure below:
   If the sub-account you are using has not performed this operation, you may encounter problems using CFS related functions.



[redacted] Sub-user

Account ID [redacted]

Remarks -

Access Method Console access,Programming access

Tag No tag

Mobile - Changing

Email -

WeChat -

Quick Login

https://cloud.tencent.com/login/subAccount/2384822478?type=subAccount&username=brucefyang

Permissions Policy

Associate a policy to get the action permissions that the policy contains. Disassociating a policy will result in losing the action permissions in the policy. A policy inherited from a use group can be disassociated only by removing the user from the user group.

Policy Name	Description	Association Type	Policy Type	Association Time	Operation
QcloudCFSReadOnlyAccess	Read-only access to Cloud File Storage (CFS)	Associated directly	Preset Policy	2023-08-21 19:30:10	Disassociate
QcloudAccessForTCMRoleInSampleDeployment	This policy is for the TCM service role(TCM_QCSRole) t...	Associated directly	Preset Policy	2023-05-08 19:37:32	Disassociate
QcloudAccessForCMRoleInPromHostingService	This policy is for the CM service role(CM_QCSRole) to ...	Associated directly	Preset Policy	2023-05-08 19:37:32	Disassociate
QcloudTKEInnerFullAccess	Full read-write access to Tencent Kubernetes Engine(T...	Associated directly	Preset Policy	2023-05-08 19:37:32	Disassociate

Creating a VPC

Please refer to Quickly Building an IPv4 Private Network to complete the creation of a VPC.

Creating CFS resources

Please refer to Creating a CFS File System to complete the creation process.

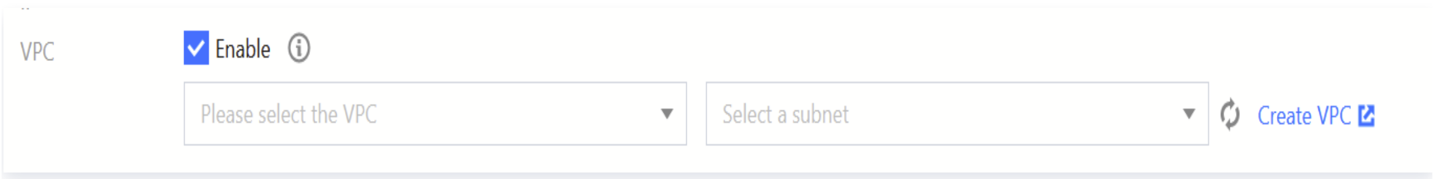
Note

Currently, SCF only supports adding CFS file systems with a network type of VPC as mount points. When creating the CFS file system, please select the same VPC as the function to ensure network connectivity.

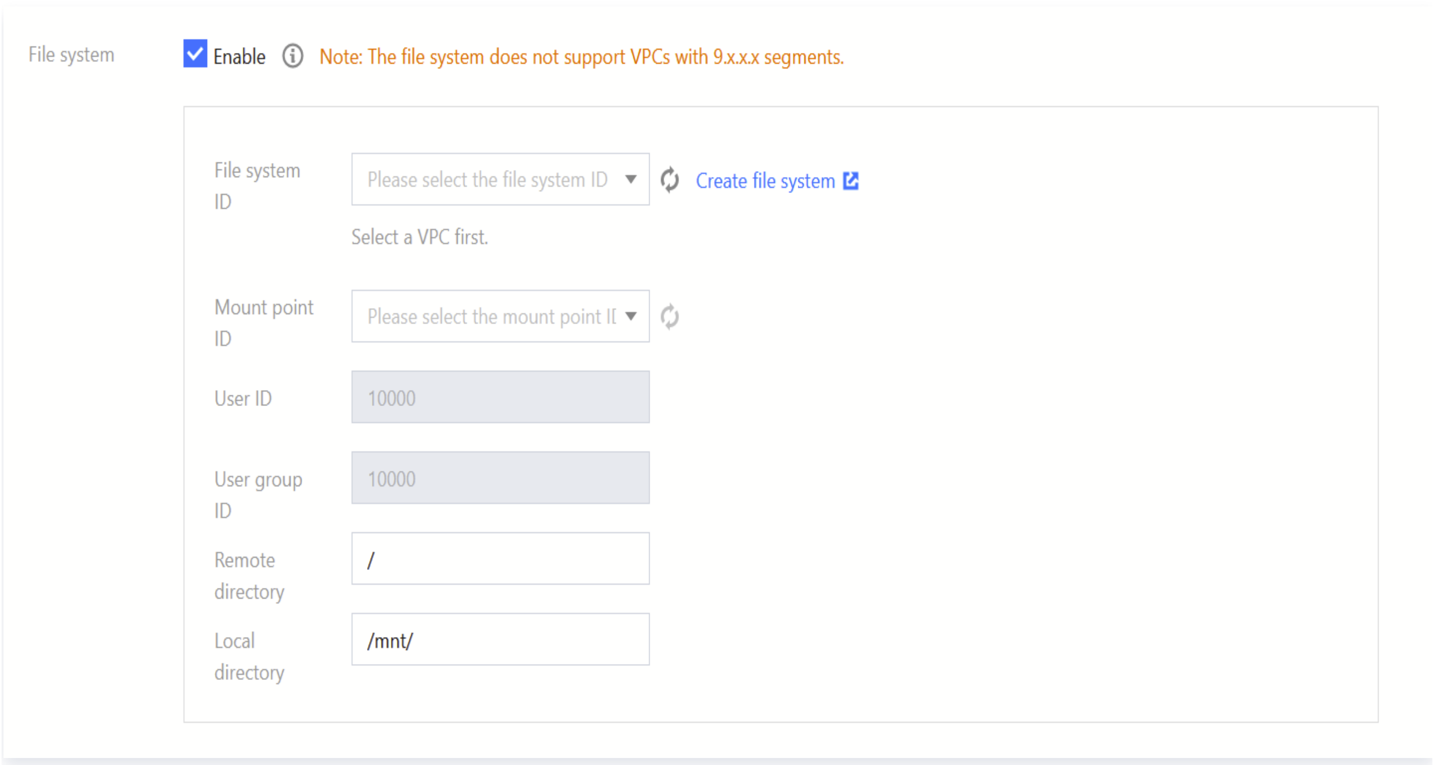
Mounting and using a CFS file system

1. Log in to the SCF console and select Function Service on the left sidebar.
2. On the **Function Service** page, select the name of the function to be configured.

3. On the "Function Management" page, under the **Function Configuration** tab, click **Edit** in the upper right corner.
4. Check **Enable** for **VPC** and select the VPC where your CFS file system resides. See below:



5. Check **Enable** for **File system** and enter the following information to mount the file system. See below:



- **User ID** and **Group ID**: These two values correspond to the user and user group in the CFS file system. The default user and user group values for cloud functions are 10000, which operate your CFS file system. Please set the permissions of the file owner and the corresponding group as needed, and ensure that your CFS file system has configured the corresponding permissions. For more details, please refer to Permission Settings.
- **Remote Directory**: This is the remote directory of the CFS file system that the cloud function needs to access. It consists of two parts: the file system and the remote directory.
- **Local Directory**: This is the mount point for the local file system. You can use a subdirectory of the `/mnt/` directory to mount the CFS file system.

- **File System ID**: Select the file system to be mounted from the drop-down list.
- **Mount Point ID**: Select the corresponding file system's mount point ID from the dropdown list.

6. Click **Save** at the bottom of the page to complete the configuration.
   You can execute 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 script to test the performance of SCF when using CFS.

Using CFS Turbo on TKE

Last updated：2023-08-29 10:41:41

Feature Overview

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

Preparations

- 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 system list and the private client installation method, refer to the Using CFS Turbo File Systems on Linux Clients document.

Instructions

Using kubectl to connect to a cluster

You can refer to Connecting to TKE to configure kubectl for managing TKE clusters.

Creating a pod for mounting Turbo using YAML files

1. Please refer to the Documentation for TKE Turbo Plugin.
2. Navigate to the `kubernetes-csi-tencentcloud/deploy/cfsturbo/kubernetes/` directory and upload the csi-node-rbac.yaml, csi-node.yaml, and csidriver-new.yaml files to the kubectl management node.
3. Navigate 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 empty.
  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 mount directory.
- host:10.0.1.16: IP of the mount point.
- path: You can adjust it based on the target subdirectory. To directly mount the root directory, enter "/".

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

- Configure RBAC and CSI plugins.

```
kubectl apply -f csi-node-rbac.yaml && kubectl apply -f csidriver-
new.yaml && kubectl apply -f csi-node.yaml
```

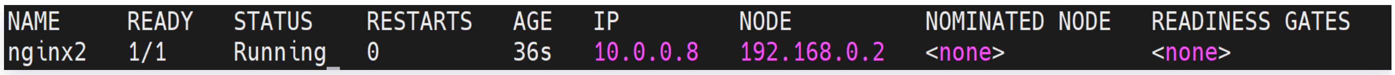
- Create PV, PVC, and pod.

```
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 view the failure reason based on the events in the TKE console.

Using CFS Turbo on TKE Serverless Cluster

Last updated：2023-08-29 10:41:48

Use Cases

Mount a CFS Turbo type storage to the Serverless container service. This component mounts the Tencent Cloud CFS Turbo file system to the workload based on a private protocol. Currently, only static configuration is supported. For more details on CFS storage types, refer to Storage Types and Performance Specifications.

Preparations

A Serverless container service of v1.14 or later has been established.

Directions

Create a file system

Create a CFS Turbo file system. For details, see Creating a File System and Mount Point.

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 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

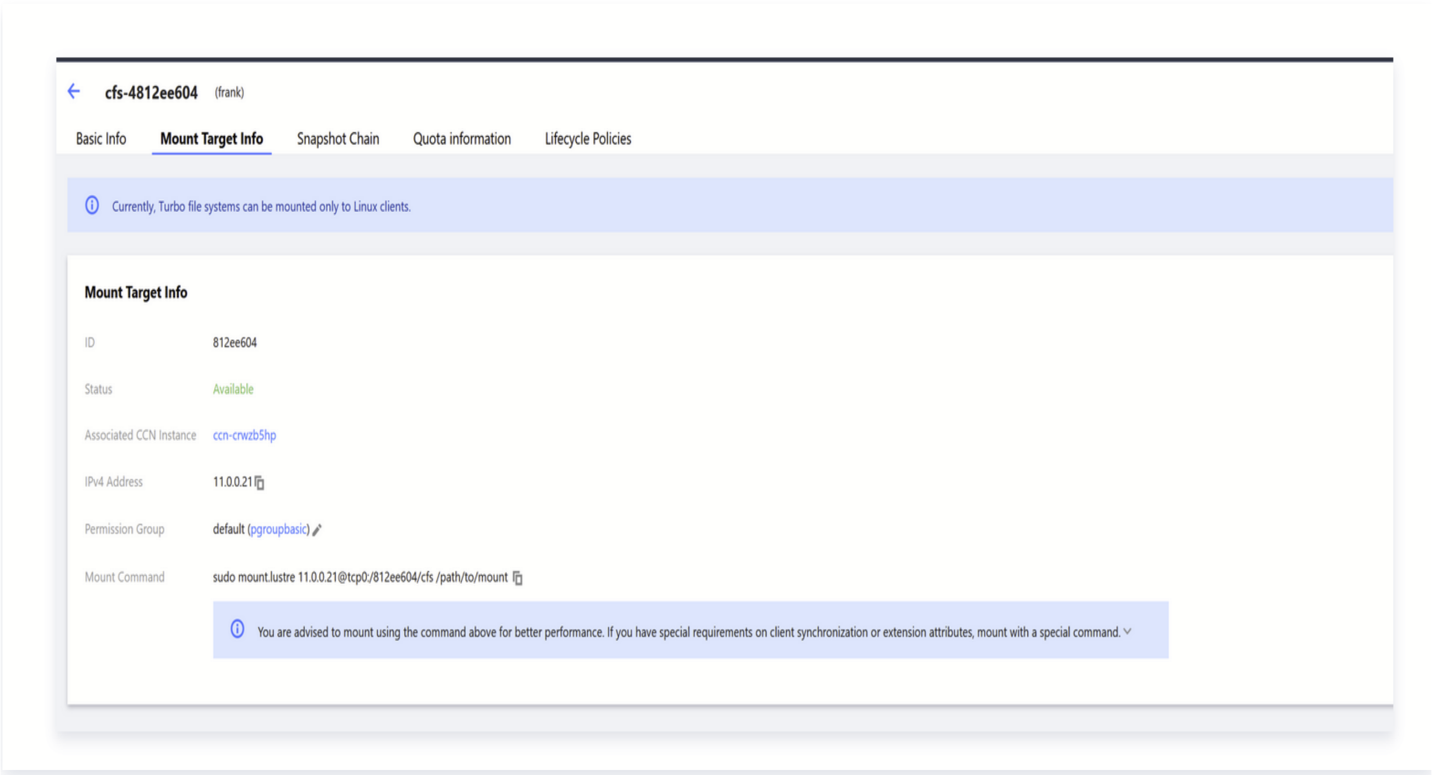
Step 1: Utilize the provided template to establish a CFS Turbo type PV.

```
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: 10.0.0.1
      Fill in according to the actual mount target IP.
      fsid: d3816815
      #Fill in according to the actual FSID
      path: /
      #cfs turbo subPath
  storageClassName: ""
```

Parameters:

- **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, which can be found in the file system mount point information.
- **spec.csi.volumeAttributes.fsid**: The fsid of the file system (not the file system ID). You can check it in the file system mount target information. It is the string between "tcp0:/" and "/cfs" in the mount command, as shown in the following figure.
- **spec.csi.volumeAttributes.path**: The subdirectory of the file system. If not specified, the default is "/" (to enhance mount performance, the plugin backend will actually locate the "/" directory to the "/cfs directory"). If you need to specify a subdirectory for mounting, make

sure that this subdirectory exists in the "/cfs" of the file system. After mounting, the workload will not be able to access the parent directory of this subdirectory. For example: path: /test, ensure that the /cfs/test directory exists in the file system.



Currently, Turbo file systems can be mounted only to Linux clients.

Mount Target Info

ID	812ee604
Status	Available
Associated CCN Instance	ccn-crwzb5hp
IPv4 Address	11.0.0.21
Permission Group	default (pgroupbasic)
Mount Command	sudo mount.lustre 11.0.0.21@tcp0:/812ee604/cfs /path/to/mount

You are advised to mount using the command above for better performance. If you have special requirements on client synchronization or extension attributes, mount with a special command.

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
```

Parameters:

- **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
```

Selecting a Network for Turbo CFS

Last updated: 2023-08-29 10:41:57

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

Feature Overview

Specified CIDR blocks 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
• CIDR block planning is separately conducted for storage systems for more efficient and convenient management of security groups. • Turbo CFS has its own CIDR block to reserve sufficient IPs for future scaling 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

You can select 10/11/30/172/192 CIDR blocks to create a CCN instance. It is recommended that Turbo CFS be assigned 11/30 blocks, which are server-side blocks and do not occupy business IPs.

VPC

Feature Overview

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

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 scaling, subnet IPs may be insufficient. • VPC IPs will be occupied, and the storage is bound to the VPC, which hinders network isolation.

Copying Data

Last updated: 2023-08-29 10:42:09

Use Cases

During your use of CFS, you may need to copy data in many scenarios. This document provides recommended solutions for fast data copying in Linux operating systems.

- **Scenario One: Data synchronization between CFS file systems, different directories within a file system, and between a file system and a cloud disk.**
  In daily use, data copying is frequently required. The above three scenarios at the Linux operating system level are all data copying between different directories, and the operation methods are similar. Normally, you can execute the cp command for simple and quick copying. However, if it involves a large number of files, the single-threaded operation mode of cp will seriously slow down the copying progress. In this case, it is recommended to use the method recommended in this document for copying, which can achieve self-concurrent acceleration.
- **Scenario Two: Data synchronization between the file system and object storage.**
  When it involves accessing storage or importing data via the public network, it is recommended to use object storage as a transit. You can use various basic tools provided by object storage to upload and download data. The recommended tools are as follows: COSBrowser, COSCMD.

Notes

These operation steps are intended for use in Scenario One. For Scenario Two, please refer to the usage pages of COSBrowser and COSCMD in Object Storage.

Prerequisites

There are source and destination directories that can be migrated on the cloud server.

Note
CFS supports cross-VPC access. You can connect multiple VPCs via Cloud Connect Network (CCN) and then mount them for access.

Instructions

1. Download and install the Rclone tool.

```
wget https://downloads.rclone.org/v1.53.4/rclone-v1.53.4-linux-
amd64.zip --no-check-certificate
```

```
unzip rclone-v1.53.4-linux-amd64.zip
chmod 0755 ./rclone-*/rclone
cp ./rclone-*/rclone /usr/bin/
rm -rf ./rclone-*
```

2. Run the following command to copy data:

```
rclone copy /mnt/src /mnt/dst -Pvv --transfers 32 --checkers 64 --
links --create-empty-src-dirs
```

This tool does not copy metadata information such as owner, ctime, and atime, by default. If you need to copy metadata information, use the `rsync` command.

Note

The parameters are described as follows; the number of transfers and checkers can be configured according to the system specifications:

- transfers: The number of concurrent file transfers (recommended to be no more than twice the number of cores).
- checkers: The number of concurrent scans of local files (recommended to be no more than twice the number of cores).
- P: Real-time display of the progress of data copying (the progress is refreshed every 500 ms. If `P` is not added to the command, the progress is refreshed every minute).
- links: Soft links for copying.
- vv: Print copying logs.
- create-empty-src-dirs: Copy empty directories.

3. After data copying is complete, you can view logs to check the results for different files.