

Hyper Computing Cluster

Operation Guide

Product Documentation



Copyright Notice

©2013-2024 Tencent Cloud. All rights reserved.

Copyright in this document is exclusively owned by Tencent Cloud. You must not reproduce, modify, copy or distribute in any way, in whole or in part, the contents of this document without Tencent Cloud's the prior written consent.

Trademark Notice



All trademarks associated with Tencent Cloud and its services are owned by Tencent Cloud Computing (Beijing) Company Limited and its affiliated companies. Trademarks of third parties referred to in this document are owned by their respective proprietors.

Service Statement

This document is intended to provide users with general information about Tencent Cloud's products and services only and does not form part of Tencent Cloud's terms and conditions. Tencent Cloud's products or services are subject to change. Specific products and services and the standards applicable to them are exclusively provided for in Tencent Cloud's applicable terms and conditions.

Contents

Operation Guide

- Managing Hyper Computing Cluster

- Installing nvidia-fabricmanager Service on GPU Instance

- Installation Instructions of TCCL on GPU Instances

- Installing RDMA Millisecond-Level Monitoring Component on GPU Instances

Operation Guide

Managing Hyper Computing Cluster

Last updated : 2024-08-20 17:03:18

Overview

Hyper Computing Cluster is used to implement RDMA network isolation management of hyper computing instances. Within the same cluster, instances are interconnected with RDMA networks.

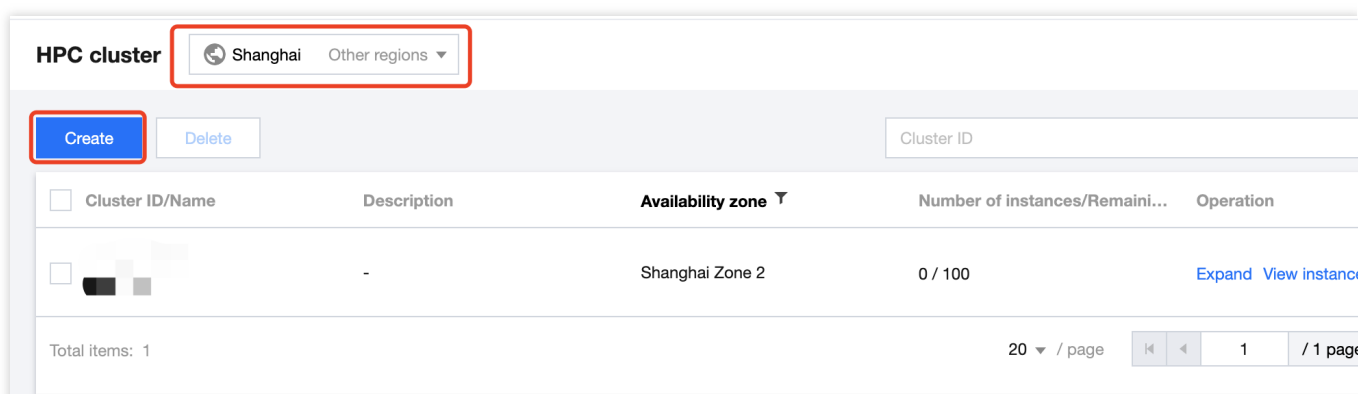
Across clusters, instances are isolated from RDMA networks.

Before creating a hyper computing **instance**, you first need to create a Hyper Computing **Cluster**. Subsequently, when instances are created, high-speed computing network interconnection of nodes within the cluster can be achieved through selecting an existing Hyper Computing Cluster.

This document describes common operations related to the Hyper Computing Cluster, such as creating, modifying, scaling out, and deleting clusters. The specific operation steps are as follows:

Creating Hyper Computing Cluster

1. Log in to the [CVM console](#), and choose **Hyper Computing Cluster** in the left sidebar.
2. On the **Hyper Computing Cluster** list page, select **Region** as needed.
3. Click **Create**.



4. In the pop-up **Create cluster** window, select and fill in the **Availability zone**, **Cluster name**, and **Cluster Description**.

Create cluster

Availability zone * Shanghai Zone 2 Shanghai Zone 5

Cluster name *

You can enter 60 more characters.

Cluster Description

You can enter 256 more characters.

5. After the information is confirmed, click OK and wait for the cluster creation to be completed.

Modifying Hyper Computing Cluster Information

1. Log in to the [CVM console](#), and choose **Hyper Computing Cluster** in the left sidebar.
2. On the **Hyper Computing Cluster** page, select



on the right side of the cluster name or description that needs to be modified, as shown in the following figure.

<input type="button" value="Create"/>	<input type="button" value="Delete"/>	<input type="text" value="Cluster ID"/>	
<input type="checkbox"/> Cluster ID/Name	Description	Availability zone	Number of instances/Remaini...
<input type="checkbox"/> hpc- test cluster	cluster name	Shanghai Zone 8	0 / 100

3. In the pop-up **Modify Name** or **Modify Description** window, enter the new cluster name and description, click **OK** to complete the operation.

Rename

[Collapse](#)

ID/Name	Availability zone
hpc-test cluster	Shanghai Zone

Name

You can enter 48 more characters.

Scaling out Hyper Computing Cluster

1. Log in to the [CVM console](#), and choose **Hyper Computing Cluster** in the left sidebar.
2. On the **Hyper Computing Cluster** page, select the cluster that needs to be scaled out, click **Expand** to enter the instance **Purchase Page**.

<input type="checkbox"/>	Cluster ID/Name	Description	Availability zone [▼]	Number of instances/Remaini...
<input type="checkbox"/>		-	Shanghai Zone 2	0 / 100

Total items: 1 20 / page

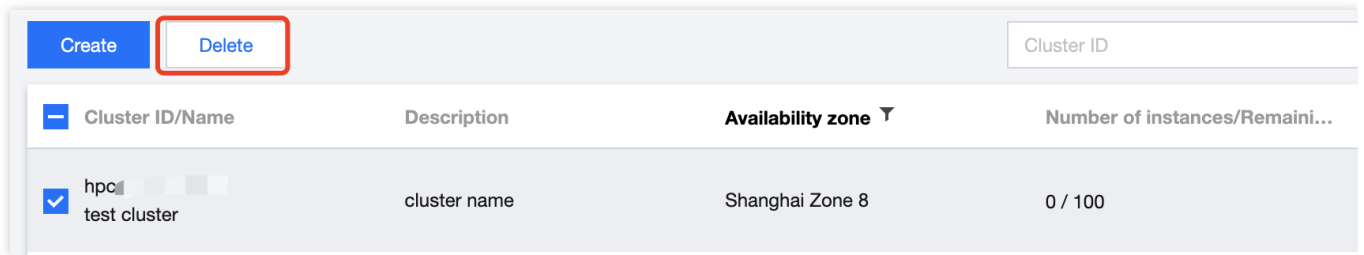
3. See [Purchase Hyper Computing Instances](#) to complete the scale-out operation.

Deleting Hyper Computing Cluster


Note:

The cluster cannot be deleted if instances have been deployed in the Hyper Computing Cluster. The cluster can be deleted only after all instances within the cluster are terminated.

1. Log in to the [CVM Console](#), and choose Hyper Computing Cluster in the left sidebar.
2. On the **Hyper Computing Cluster** page, select one or more clusters as needed, and then click **Delete**.



Cluster ID

<input type="checkbox"/>	Cluster ID/Name	Description	Availability zone 	Number of instances/Remaini...
<input checked="" type="checkbox"/>	hpc- test cluster	cluster name	Shanghai Zone 8	0 / 100

3. In the pop-up window, confirm the information and click **Confirm** to complete the operation.

Installing nvidia-fabricmanager Service on GPU Instance

Last updated : 2024-08-20 17:04:57

Overview

The Hyper Computing ClusterPNV4h instance is equipped with A100 GPUs and supports **NvLink & NvSwitch**. It requires the installation of the nvidia-fabricmanager service corresponding to the driver version to enable interconnection between GPUs. If you are using this instance, see this document to install the nvidia-fabricmanager service. Otherwise, you may not be able to use the GPU instance properly.

Directions

This document takes the driver version **470.103.01** as an example. You can follow the steps below for installation. You can replace the driver version after the `version` parameter as needed.

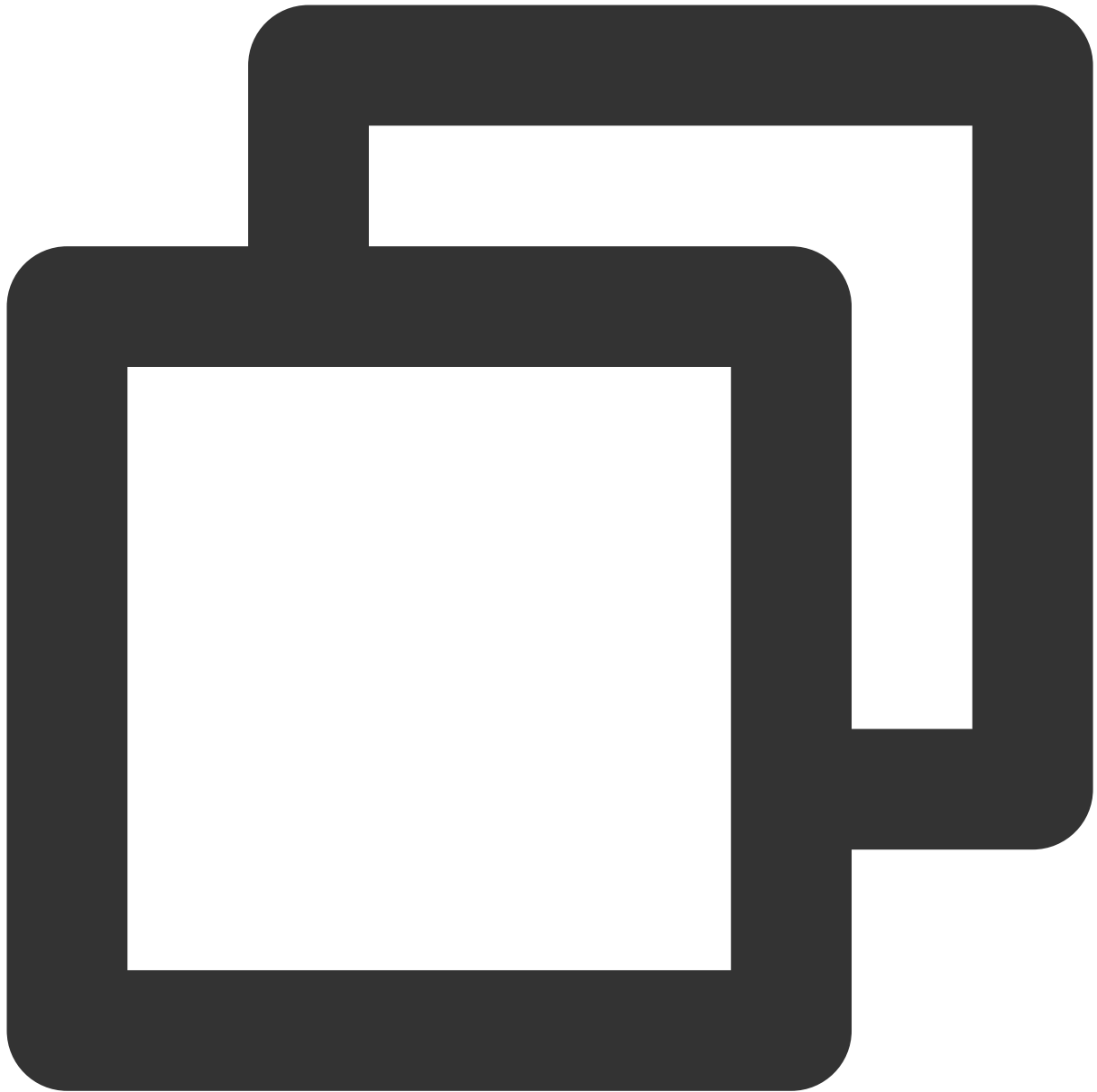
Installing nvidia-fabricmanager Service

1. Log in to the instance. For details, see [Logging in to Linux Instance \(Standard Method\)](#).
2. The installation varies by operating system. Run the corresponding command for installation.

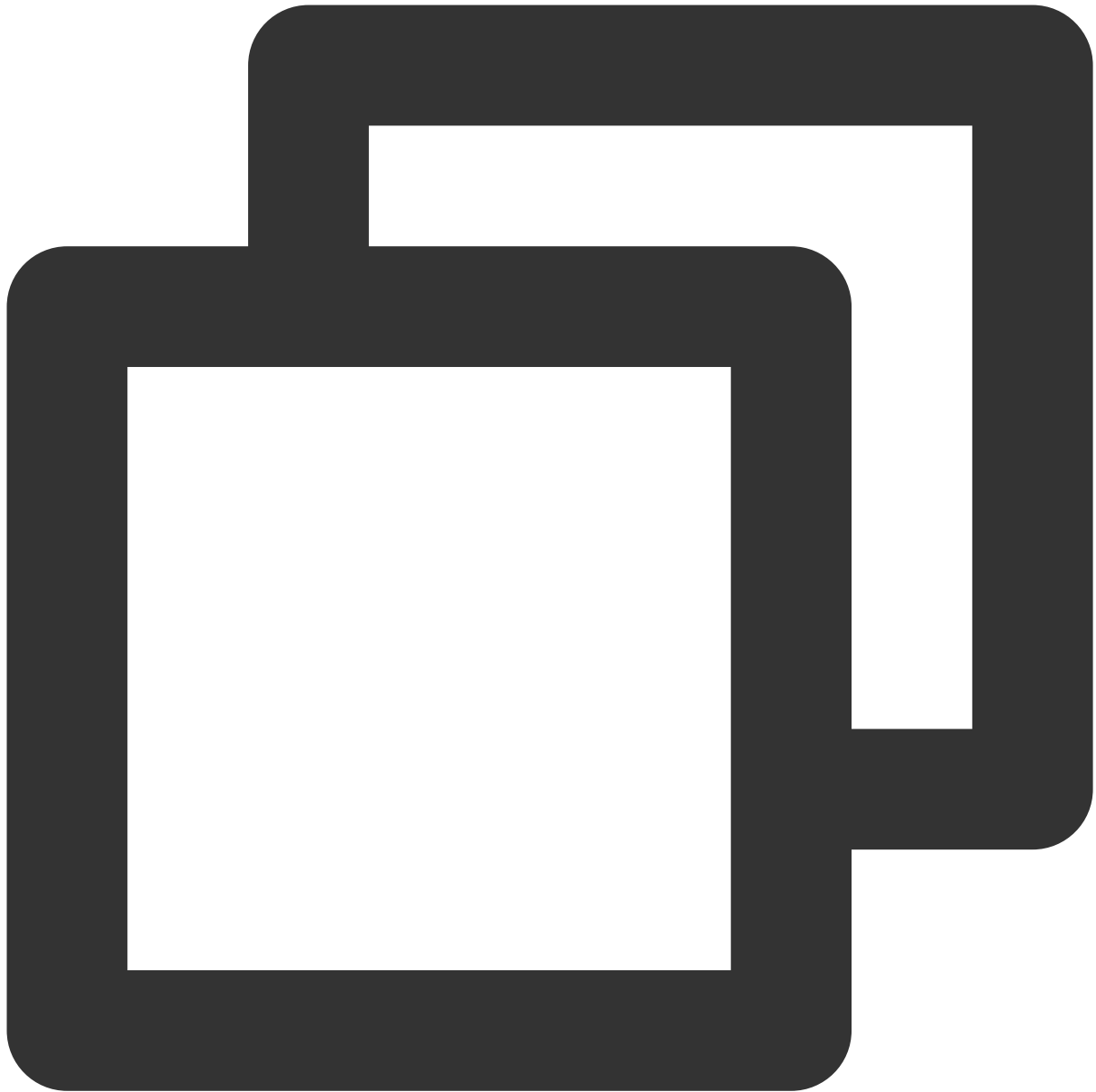
CentOS 7.x Image

Ubuntu 18.04 Image

TencentOS 2.4 Image



```
version=470.103.01
yum -y install yum-utils
yum-config-manager --add-repo https://developer.download.nvidia.com/compute/cuda/re
yum install -y nvidia-fabric-manager-${version}-1
```



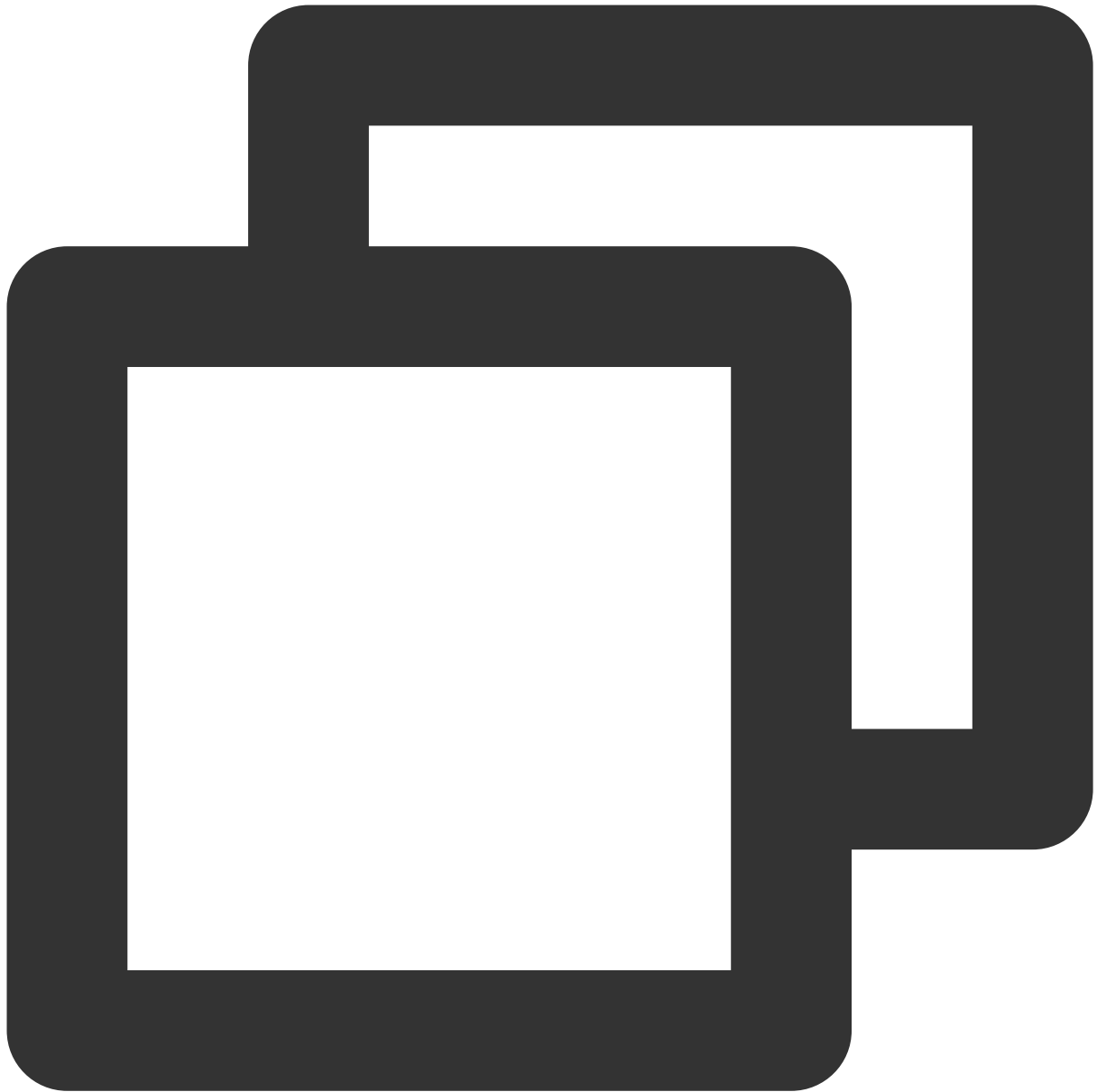
```
version=470.103.01
main_version=$(echo $version | awk -F '.' '{print $1}')
apt-get updateapt
get -y install nvidia-fabricmanager-${main_version}=${version}-*
```



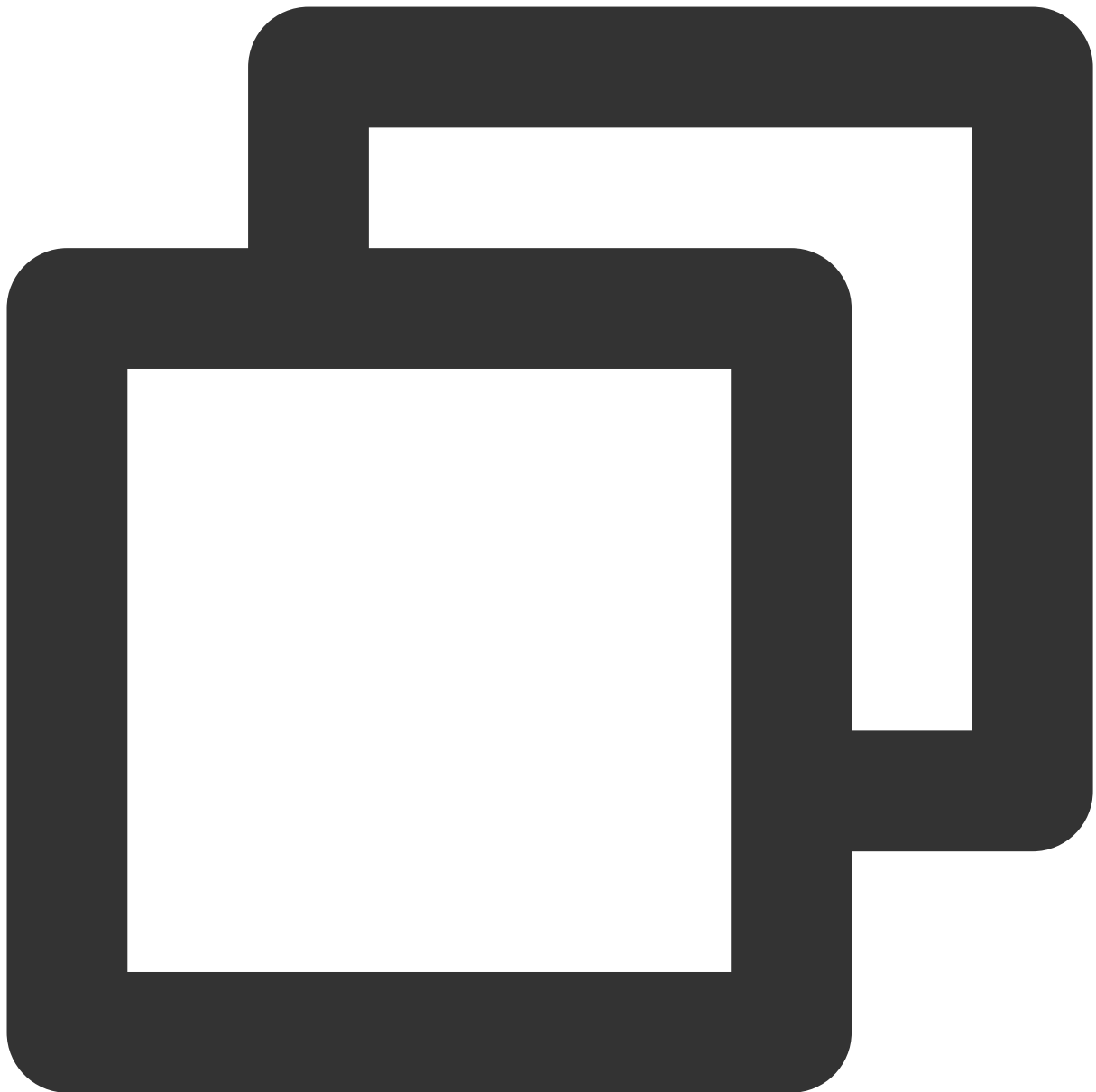
```
version=470.103.01
yum -y install yum-utils
yum-config-manager --add-repo https://developer.download.nvidia.com/compute/cuda/re
yum install -y nvidia-fabric-manager-${version}-1
```

Starting nvidia-fabricmanager Service

Run the following commands in sequence to start the service.



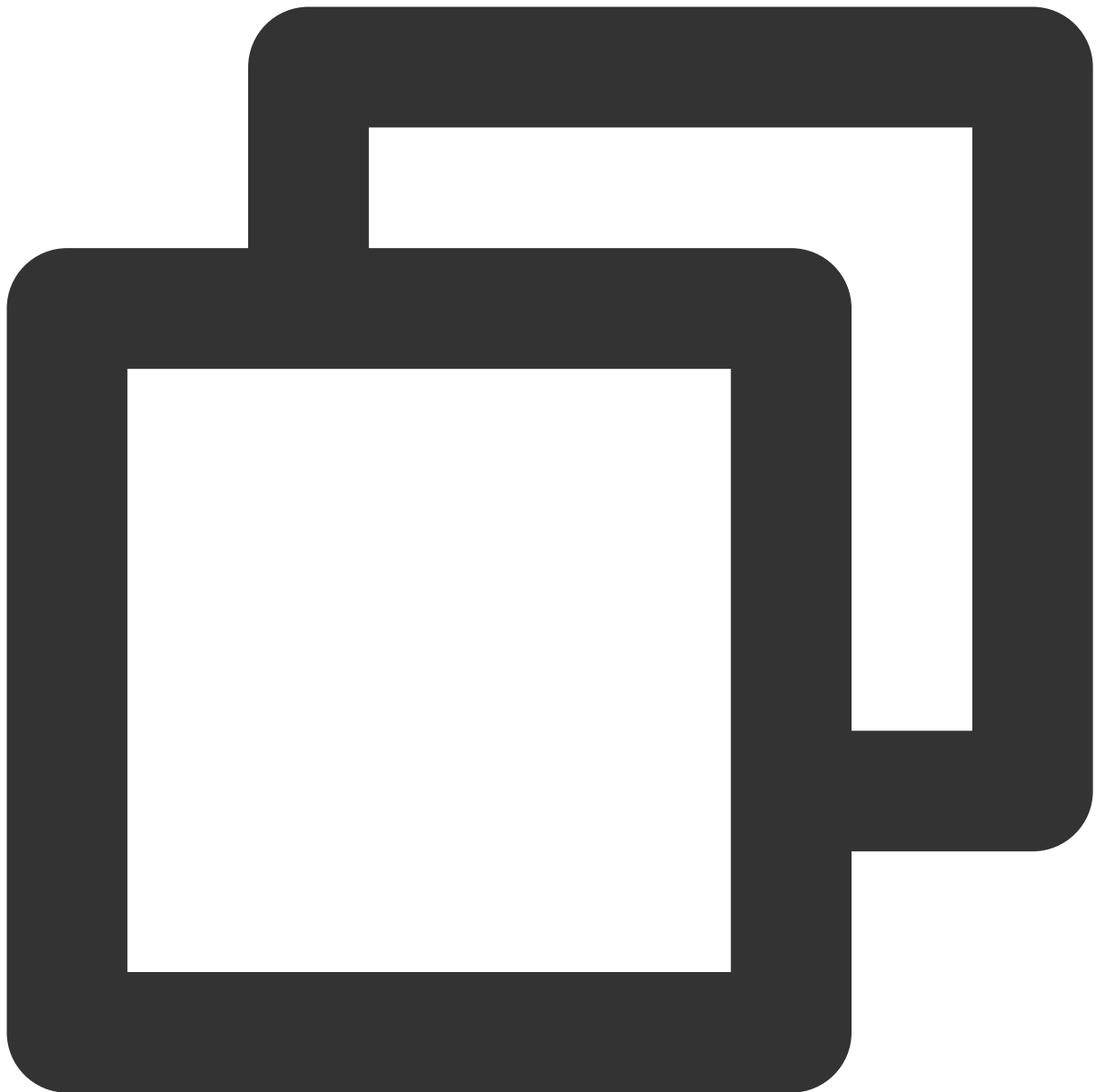
```
systemctl enable nvidia-fabricmanager
```



```
systemctl start nvidia-fabricmanager
```

Viewing nvidia-fabricmanager Service Status

Run the following command to view the service status.



```
systemctl status nvidia-fabricmanager
```

If the following information is output, the service is installed successfully.

```
root@ # systemctl status nvidia-fabricmanager
● nvidia-fabricmanager.service - NVIDIA fabric manager service
   Loaded: loaded (/lib/systemd/system/nvidia-fabricmanager.service; enabled; vendor
   Active: active (running) since Tue 2022-02-22 21:46:13 CST; 39s ago
 Main PID: 45212 (nv-fabricmanage)
   Tasks: 18 (limit: 29491)
   CGroup: /system.slice/nvidia-fabricmanager.service
           └─45212 /usr/bin/nv-fabricmanager -c /usr/share/nvidia/nvswitch/fabricman

Feb 22 21:45:59 systemd[1]: Starting NVIDIA fabric manager service.
Feb 22 21:46:13 nv-fabricmanager[45212]: Successfully configured all
Feb 22 21:46:13 systemd[1]: Started NVIDIA fabric manager service.
```

Installation Instructions of TCCL on GPU Instances

Last updated : 2024-08-20 17:06:38

Introduction of TCCL

Tencent Collective Communication Library (TCCL) is a high-performance customized acceleration communication library designed for Tencent Cloud's StarPulse network architecture. The main feature leverages the StarPulse network hardware architecture to provide more efficient network communication performance for large-scale AI model training, along with intelligent operations and maintenance capabilities for rapid perception and self-healing of network failures. TCCL has been expanded and optimized based on open source NCCL code, ensuring full compatibility with NCCL's features and usage methods. TCCL currently supports main features including:

Dynamic aggregation optimization of dual network interfaces maximizes the performance of bonding devices.

Global Hash Routing enables load balancing to avoid congestion.

Topology affinity traffic scheduling minimizes traffic detours.

Overview

This document describes how to configure the TCCL acceleration communication library in the Tencent Cloud environment to improve multi-machine multi-card communication performance in the Tencent Cloud RDMA environment. In large-scale model training scenarios, TCCL is expected to increase the bandwidth utilization rate by approximately 50% compared with the open source NCCL scheme.

Directions

Environment Preparations

1. Create GPU Hyper Computing ClusterPNV4sne or GPU Hyper Computing ClusterPNV4sn Hyper Computing Cluster instances, which support 1.6 Tbps and 800 Gbps RDMA networks respectively.
2. Install the GPU driver and [nvidia-fabricmanager service](#) for GPU instances.

Note:

TCCL operating software environment requires glibc version 2.17 or later and CUDA version 10.0 or later.

Selecting Installation Methods

TCCL currently supports three installation methods. You can select the installation method suitable for business scenarios as required.

TCCL communication library + compile and install pytorch

TCCL communication library + pytorch communication plugins

NCCL plugins + sorted IP list

Note:

Since most large-scale model training is based on the Pytorch framework, we will take Pytorch as an example.

The comparison of three access schemes for TCCL is shown in the following table:

Installation Methods	Method 1: compile and install Pytorch.	Method 2: install Pytorch communication plugins.	(Recommended) Method 3: install NCCL communication plugins.
Usage Steps	Install TCCL. Recompile and install Pytorch.	Install Pytorch communication plugins. Modify the distributed communication backend.	Install NCCL plugins. Modify the startup script.
Advantage	No intrusion into business code.	Easy to install.	Easy to install.
Disadvantage	Require to recompile and install Pytorch. Have requirements for the software environment.	Require to modify the business code. Have requirements for the software environment.	Require to update the sorted list after scaling out cluster nodes.
Software Dependency on the Environment	Corresponding to NCCL version 2.12. Require glibc version 2.17 or later. Require CUDA version 10.0 or later.	Current installation package only supports Pytorch 1.12 Require glibc version 2.17 or later. Require CUDA version 10.0 or later.	Install NCCL.

If your machine resources and model training scenarios are relatively fixed, it is recommended to use Method 3, which is compatible with different NCCL and CUDA versions, and easy to install and use without modifying business code or recompiling pytorch.

If your resources need to be provided to different business teams or frequently require scale-out, it is recommended to use the first two methods, which do not require algorithm personnel or scheduling frameworks to deliberately perceive the network topology information of machines.

If you do not want to adapt the business code, you can use Method 1, which only requires recompiling the pytorch framework.

Configuring the TCCL Environment and Verifying

Method 1: compile and install Pytorch.

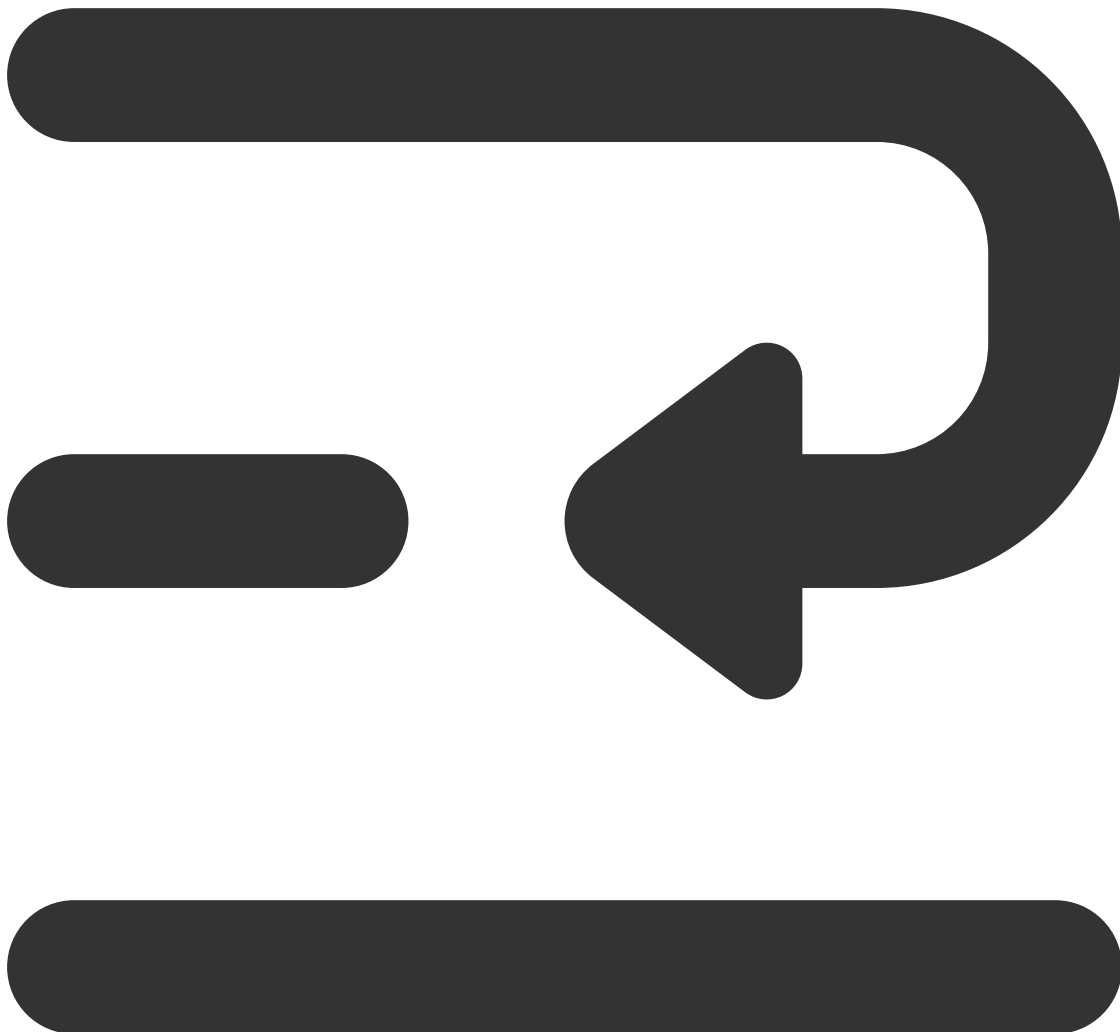
Method 2: install Pytorch communication plugins.

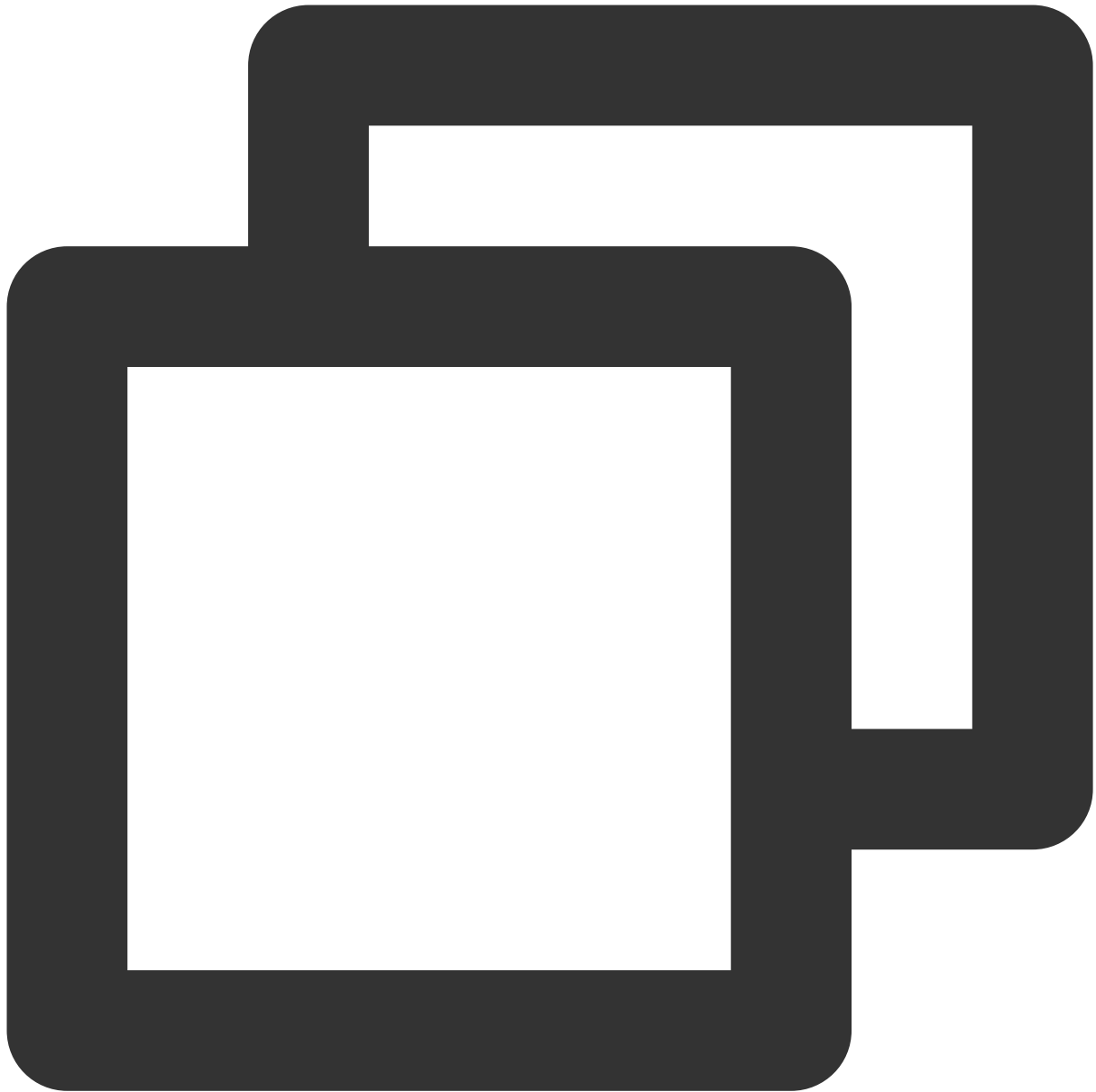
(Recommended) Method 3: install NCCL plugins

As the community pytorch connects to the NCCL communication library statically by default, TCCL cannot be used by replacing the shared libraries.

1. Install TCCL.

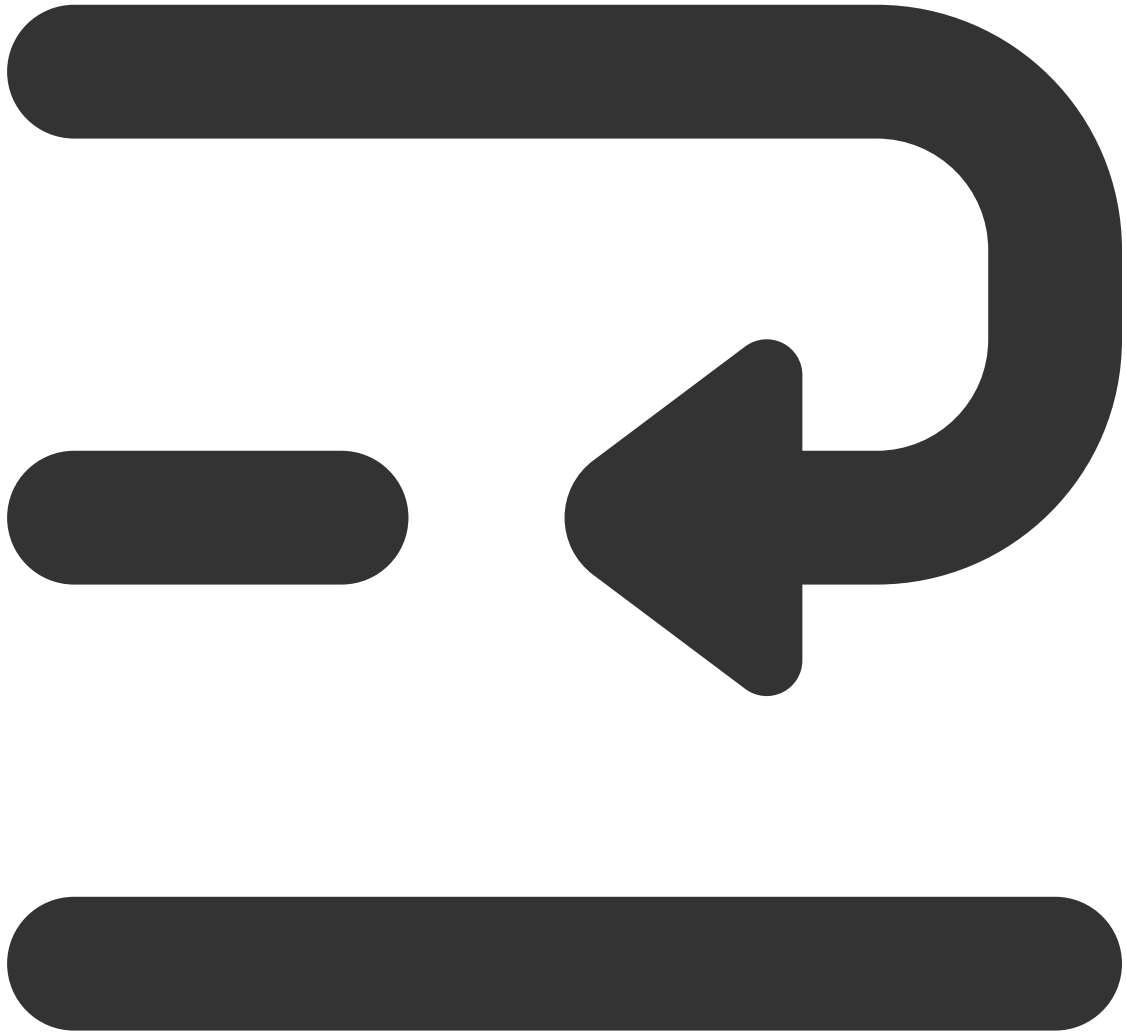
Take Ubuntu 20.04 as an example, you can use the following commands to install. After installation, TCCL will be located in the `/opt/tencent/tccl` directory.





```
# Uninstall the existing tccl versions and nccl plugins.  
dpkg -r tccl && dpkg -r nccl-rdma-sharp-plugins  
  
# Download and install tccl v1.5 version.  
wget https://taco-1251783334.cos.ap-shanghai.myqcloud.com/tccl/TCCL_1.5-ubuntu.20.0
```

If you use CentOS or TencentOS, see the following steps for installation:

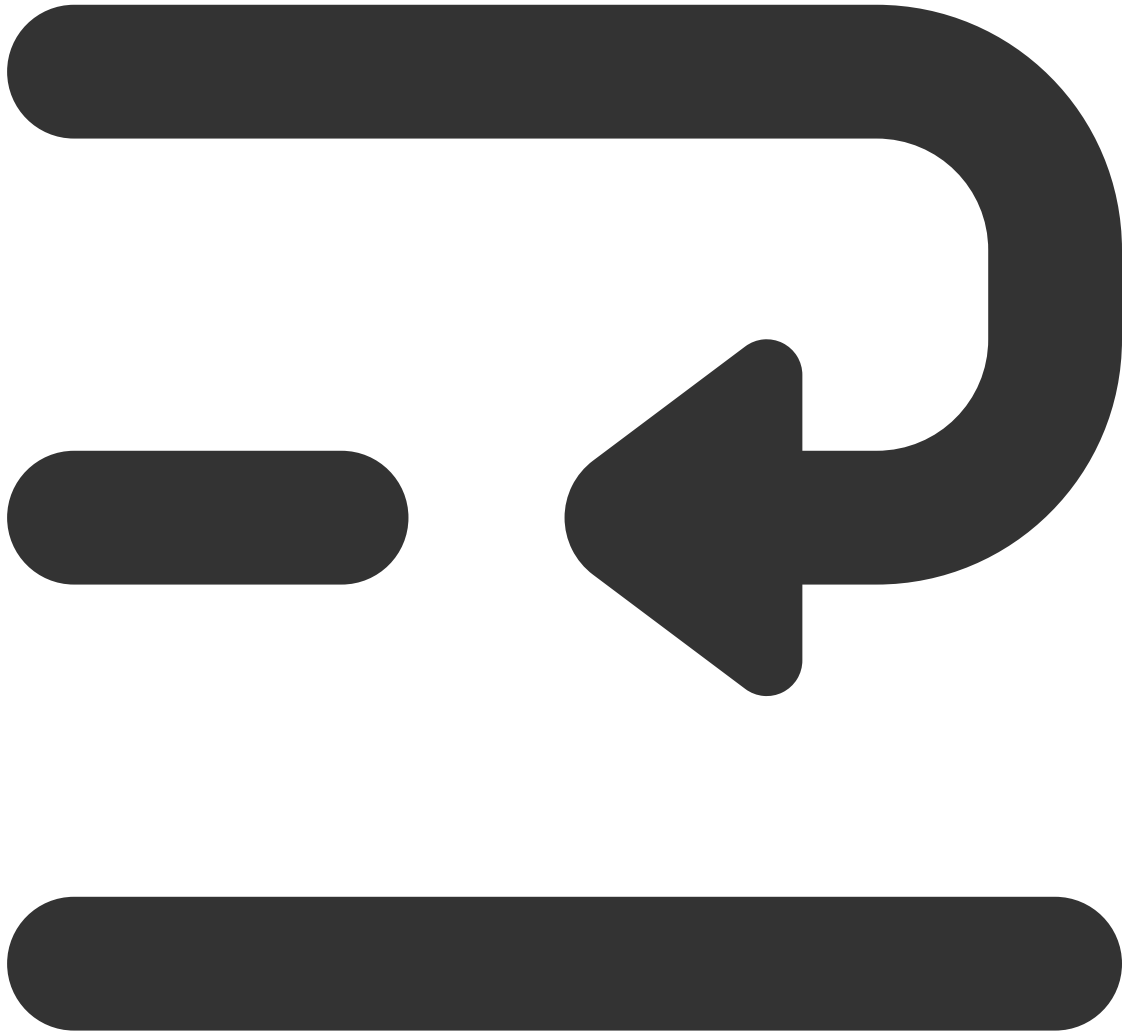


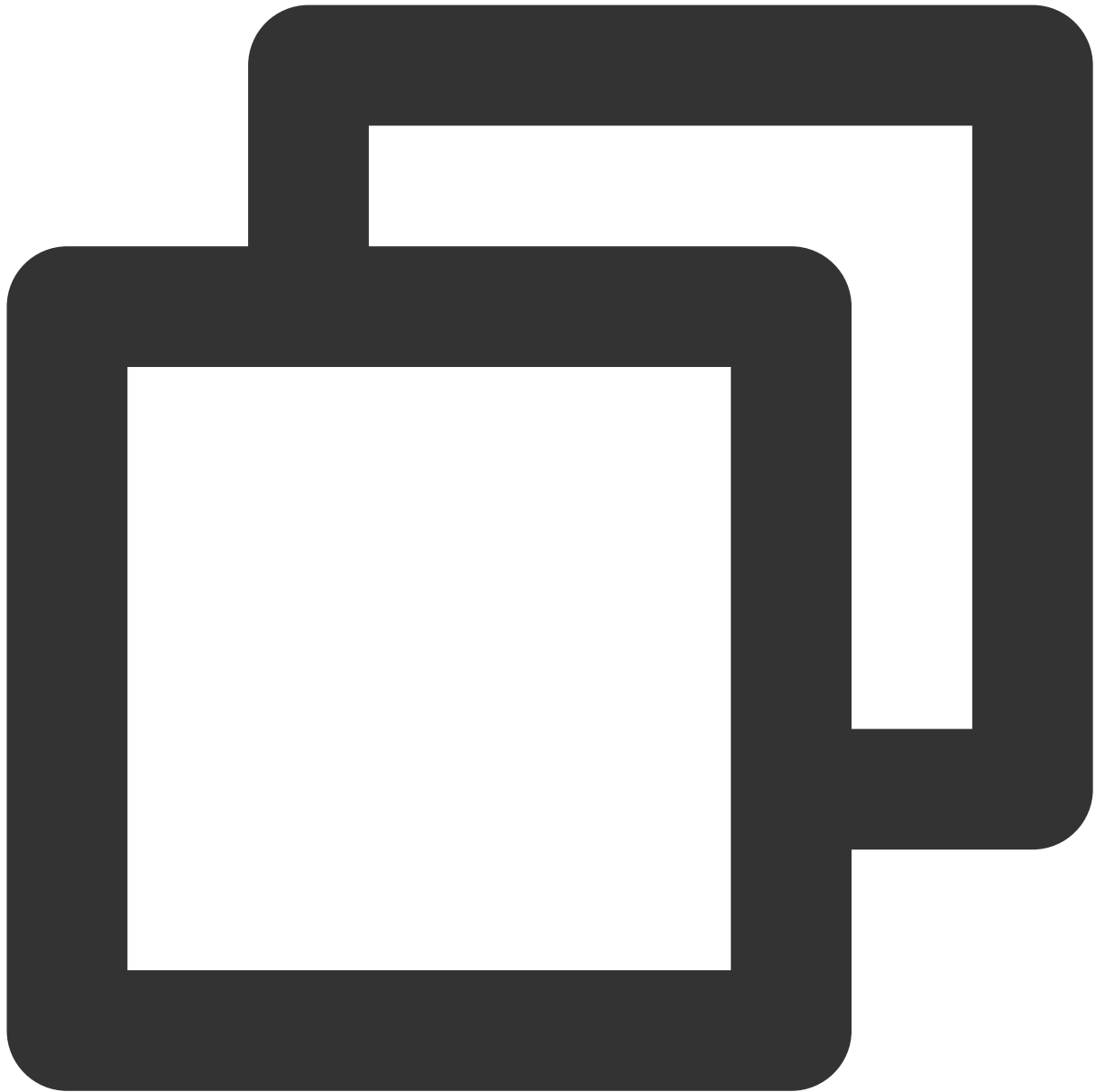


```
# Uninstall the existing tccl versions and nccl plugins.  
rpm -e tccl && rpm -e nccl-rdma-sharp-plugins-1.0-1.x86_64  
  
# Download tccl v1.5 version.  
wget https://taco-1251783334.cos.ap-shanghai.myqcloud.com/tccl/tccl-1.5-1.t12.x86_6
```

2. Recompile and install Pytorch.

The following is an example of Pytorch source code installation. See the [Official Pytorch Installation Description](#) for details.





```
#!/bin/bash

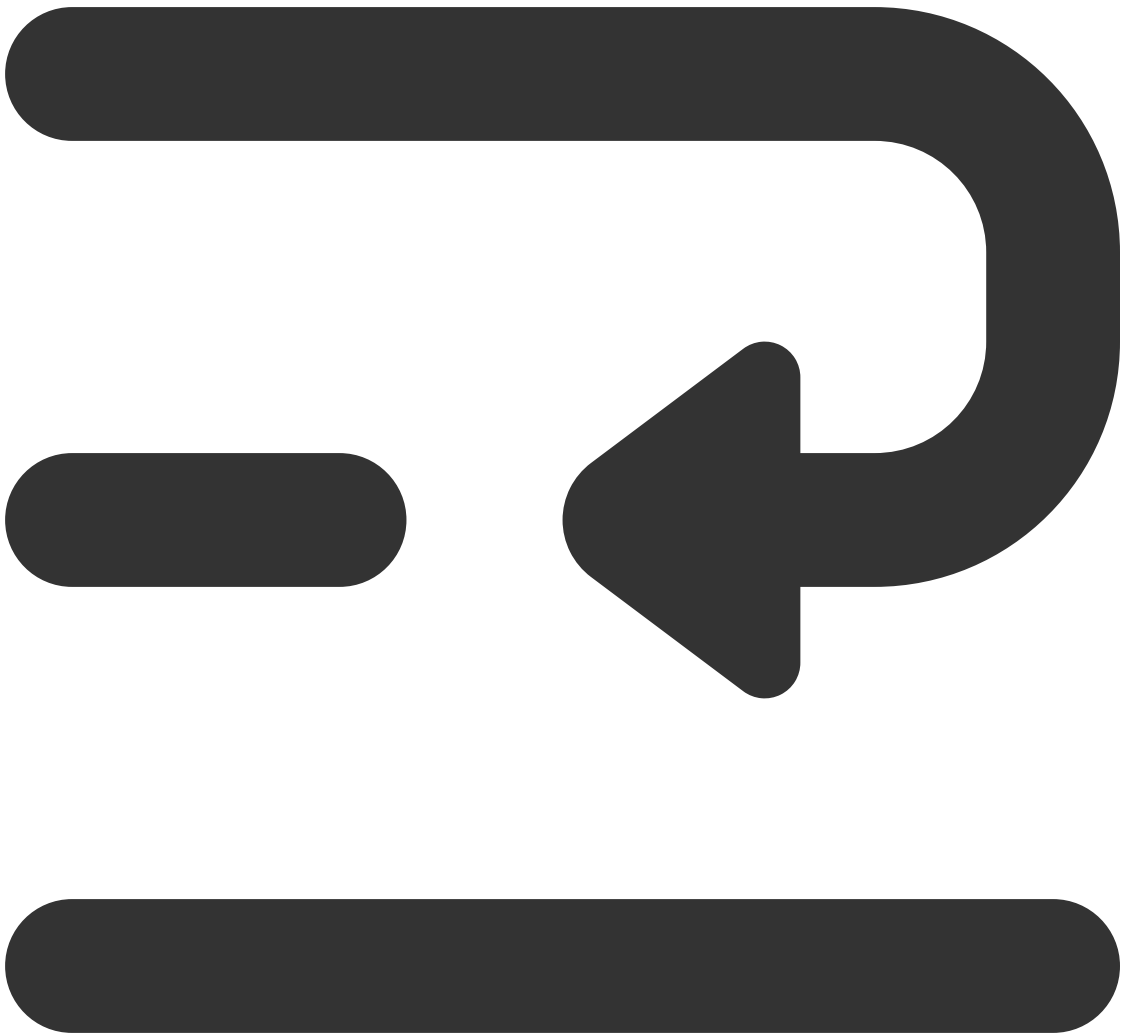
# Uninstall the current version.
pip uninstall -y torch

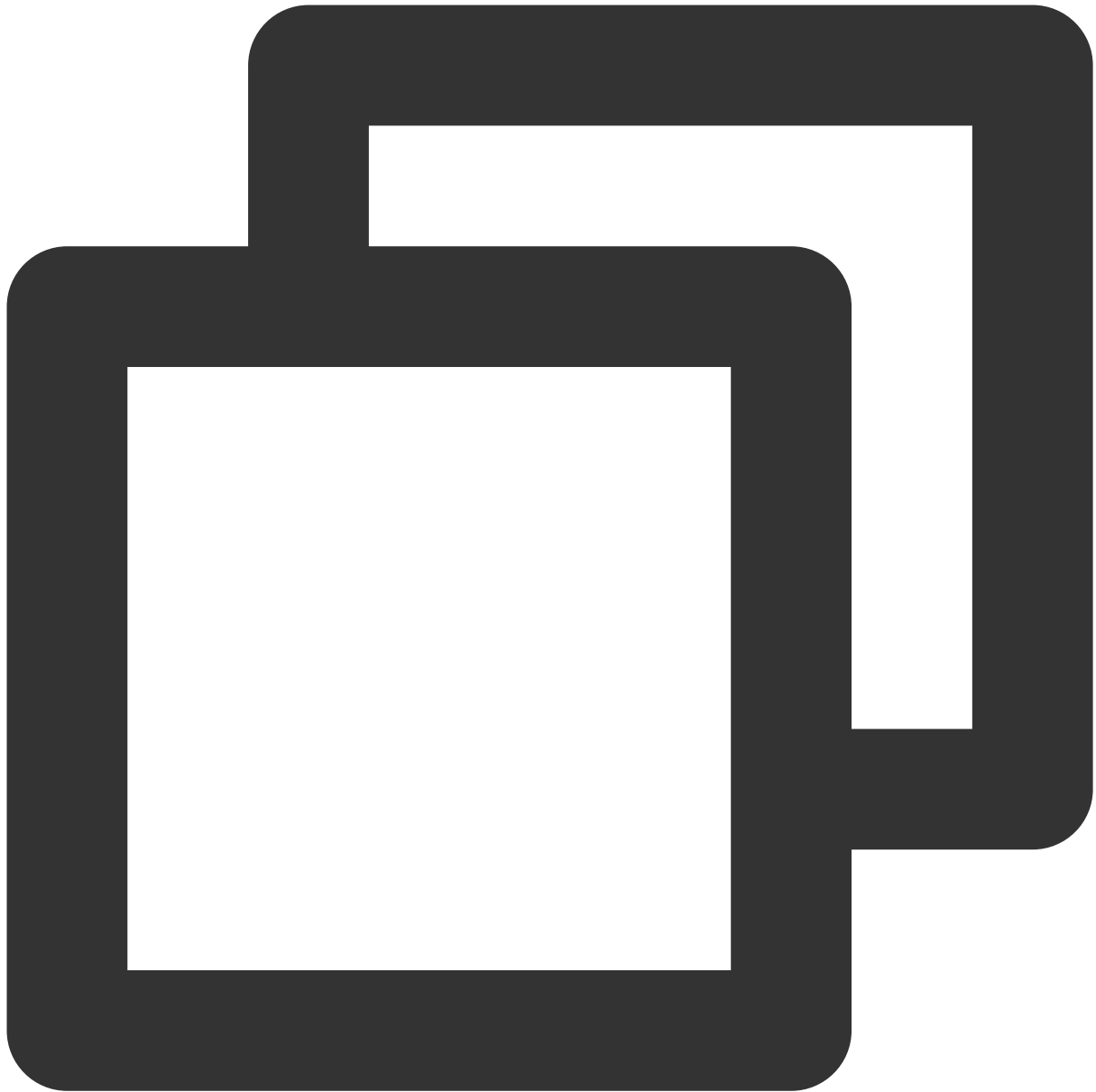
# Download pytorch source code.
git clone --recursive https://github.com/pytorch/pytorch
cd pytorch

# <!Important> Configure the installation path of TCCL.
export USE_SYSTEM_NCCL=1
```

```
export NCCL_INCLUDE_DIR="/opt/tencent/tccl/include"  
export NCCL_LIB_DIR="/opt/tencent/tccl/lib"  
  
# See the official website to add other compilation options .  
  
# Install the development environment.  
python setup.py develop
```

3. Configure TCCL environment variables.





```
export NCCL_DEBUG=INFO
export NCCL_SOCKET_IFNAME=eth0
export NCCL_IB_GID_INDEX=3
export NCCL_IB_DISABLE=0
export NCCL_IB_HCA=mlx5_bond_0,mlx5_bond_1,mlx5_bond_2,mlx5_bond_3,mlx5_bond_4,mlx5
export NCCL_NET_GDR_LEVEL=2
export NCCL_IB_QPS_PER_CONNECTION=4
export NCCL_IB_TC=160
export NCCL_IB_TIMEOUT=22
export NCCL_PXN_DISABLE=0
export TCCL_TOPO_AFFINITY=4
```

Note:

You need to enable network topology awareness through `TCCL_TOPO_AFFINITY=4`.

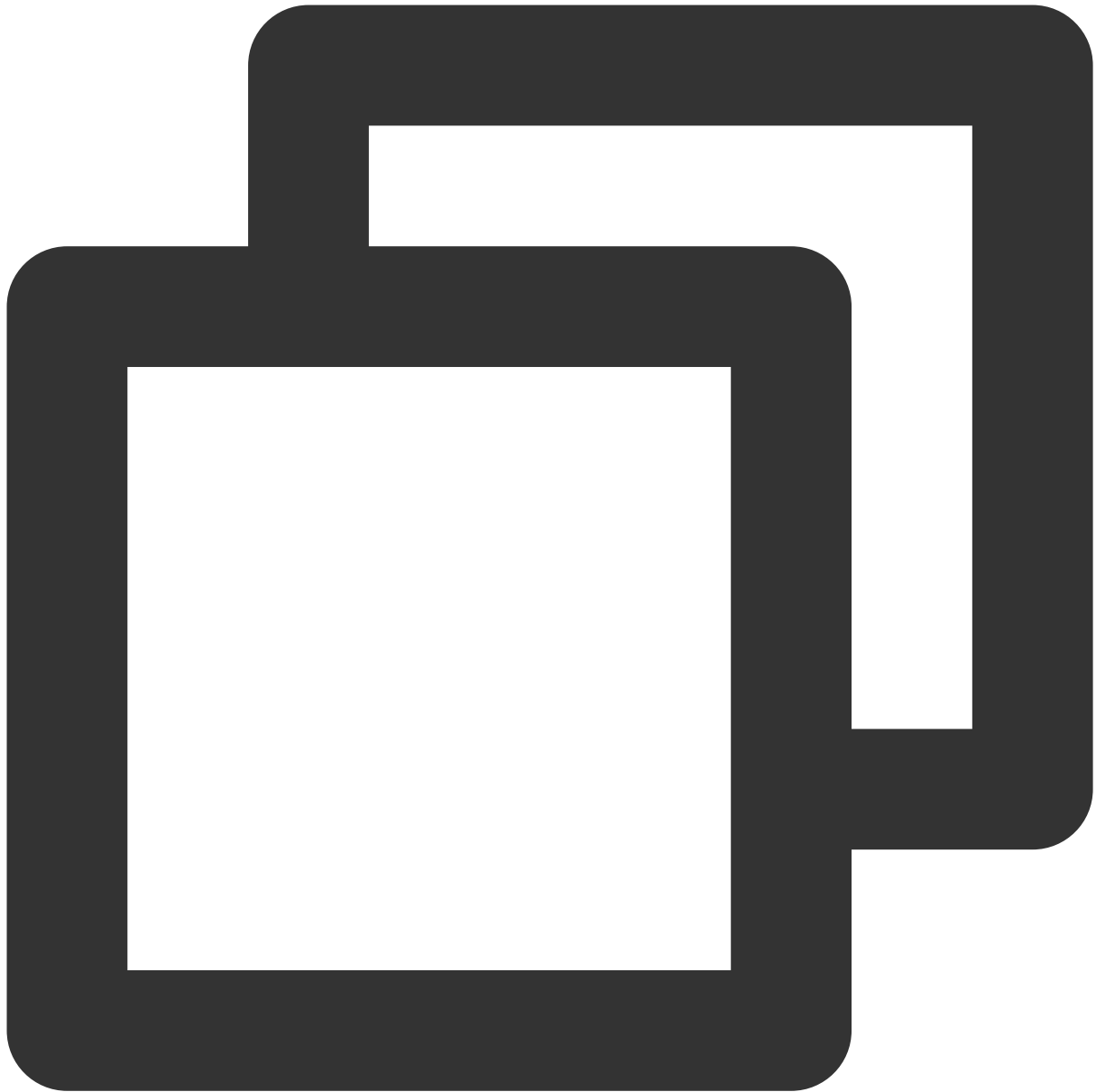
4. Verify Pytorch.

During the process of running single-machine multi-GPU or multi-machine multi-GPU training, the following log will be printed (export `NCCL_DEBUG=INFO`):

```
vm-3-17-centos:74350:74350 [0] NCCL INFO Bootstrap : Using eth0:10.100.3.17<0>
vm-3-17-centos:74350:74350 [0] NCCL INFO NET/Plugin : No plugin found (libnccl-net.so), using internal implement
vm-3-17-centos:74350:74350 [0] NCCL INFO NET/IB : Using [0]mlx5_bond_0:1/RoCE [R0]; 00B eth0:10.100.3.17<0>
vm-3-17-centos:74350:74350 [0] NCCL INFO Using network IB
NCCL version 2.12.12_TCCL_v1.5+cuda11.6
vm-3-17-centos:74352:74352 [2] NCCL INFO Bootstrap : Using eth0:10.100.3.17<0>
vm-3-17-centos:74352:74352 [2] NCCL INFO NET/Plugin : No plugin found (libnccl-net.so), using internal implement
vm-3-17-centos:74352:74352 [2] NCCL INFO NET/IB : Using [0]mlx5_bond_0:1/RoCE [R0]; 00B eth0:10.100.3.17<0>
vm-3-17-centos:74352:74352 [2] NCCL INFO Using network IB
```

5. Verify nccl-tests.

Before running `nccl-tests`, you need to export the corresponding TCCL path:



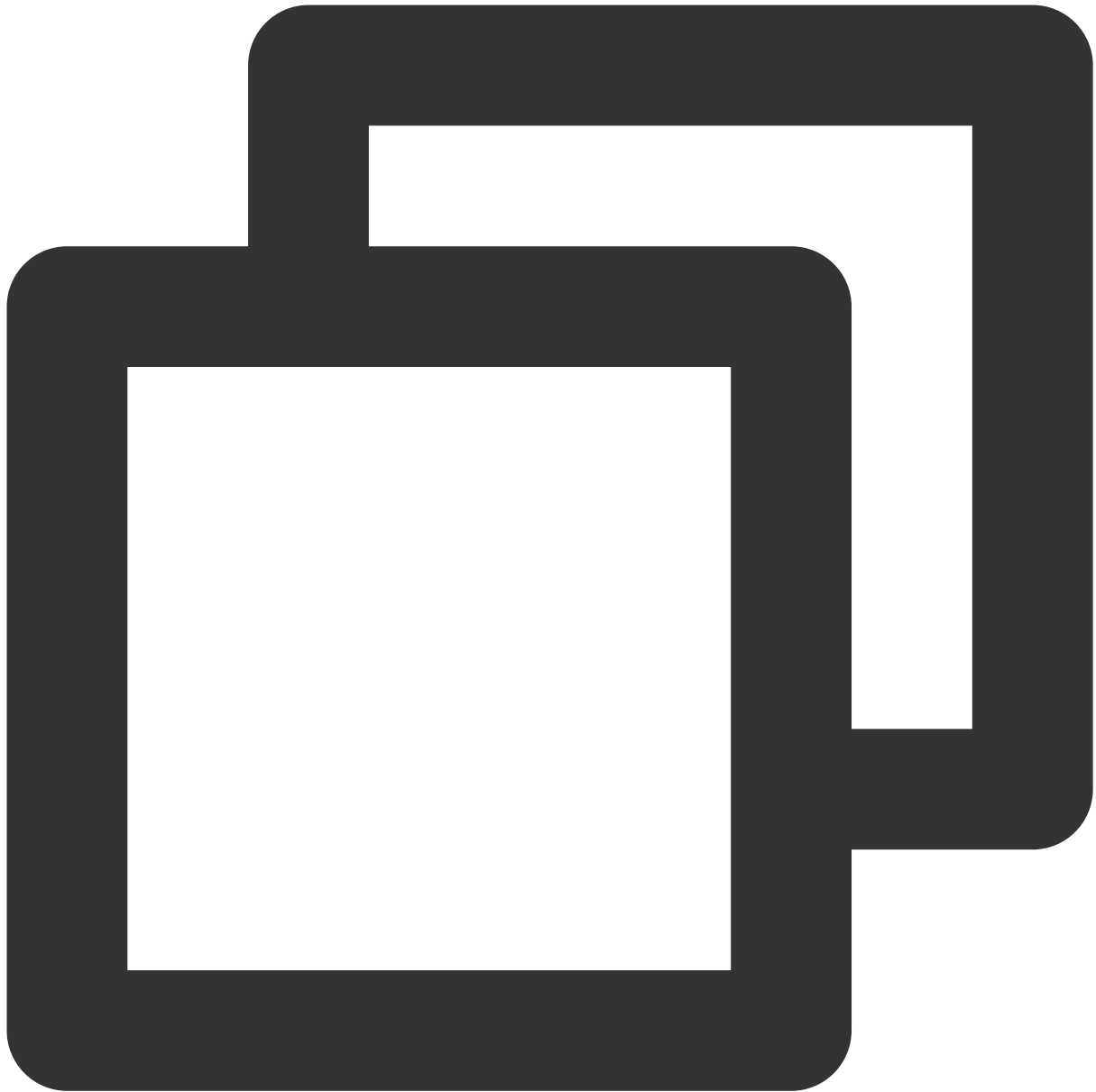
```
export LD_LIBRARY_PATH=/opt/tencent/tccl/lib:$LD_LIBRARY_PATH
```

6. Supported software versions

At present, TCCL corresponds to NCCL version 2.12, requiring glibc version 2.17 or later and CUDA version 10.0 or later. For other supported CUDA version, please contact your pre-sales manager for support.

Pytorch supports integrating third-party communication backends through plugins, so users can use TCCL communication backends without recompiling Pytorch. The API is fully compatible with NCCL. See [Introduction to Existing Communication Backends in Pytorch](#) for details.

1. Install Pytorch communication plugins.

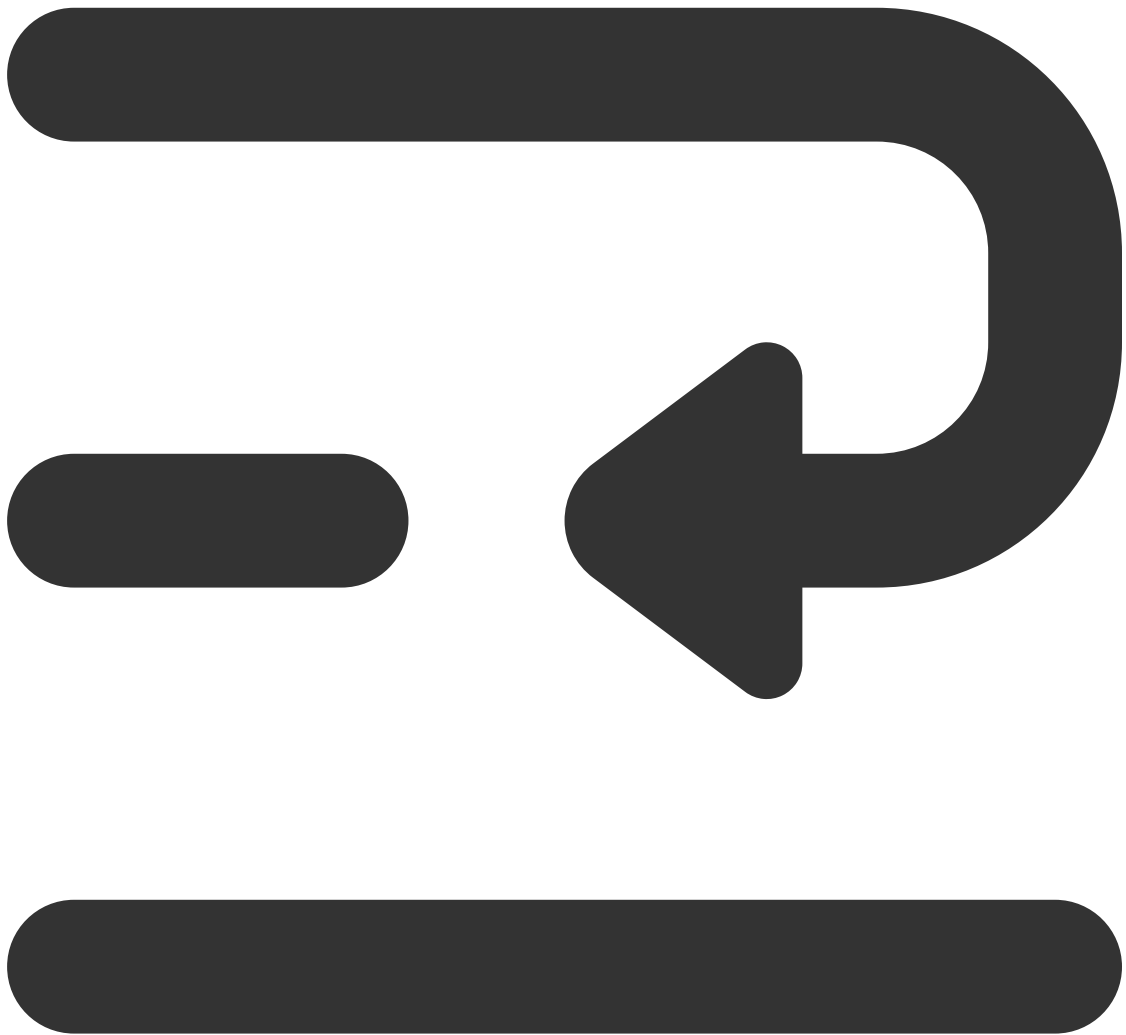


```
# Uninstall the existing tccl and NCCL plugins.
dpkg -r tccl && dpkg -r nccl-rdma-sharp-plugins

# Uninstall torch_tccl.
pip uninstall -y torch-tccl

# Install torch_tccl version 0.0.2.
wget https://taco-1251783334.cos.ap-shanghai.myqcloud.com/tccl/torch_tccl-0.0.2_pt1
```

2. Modify the business code.

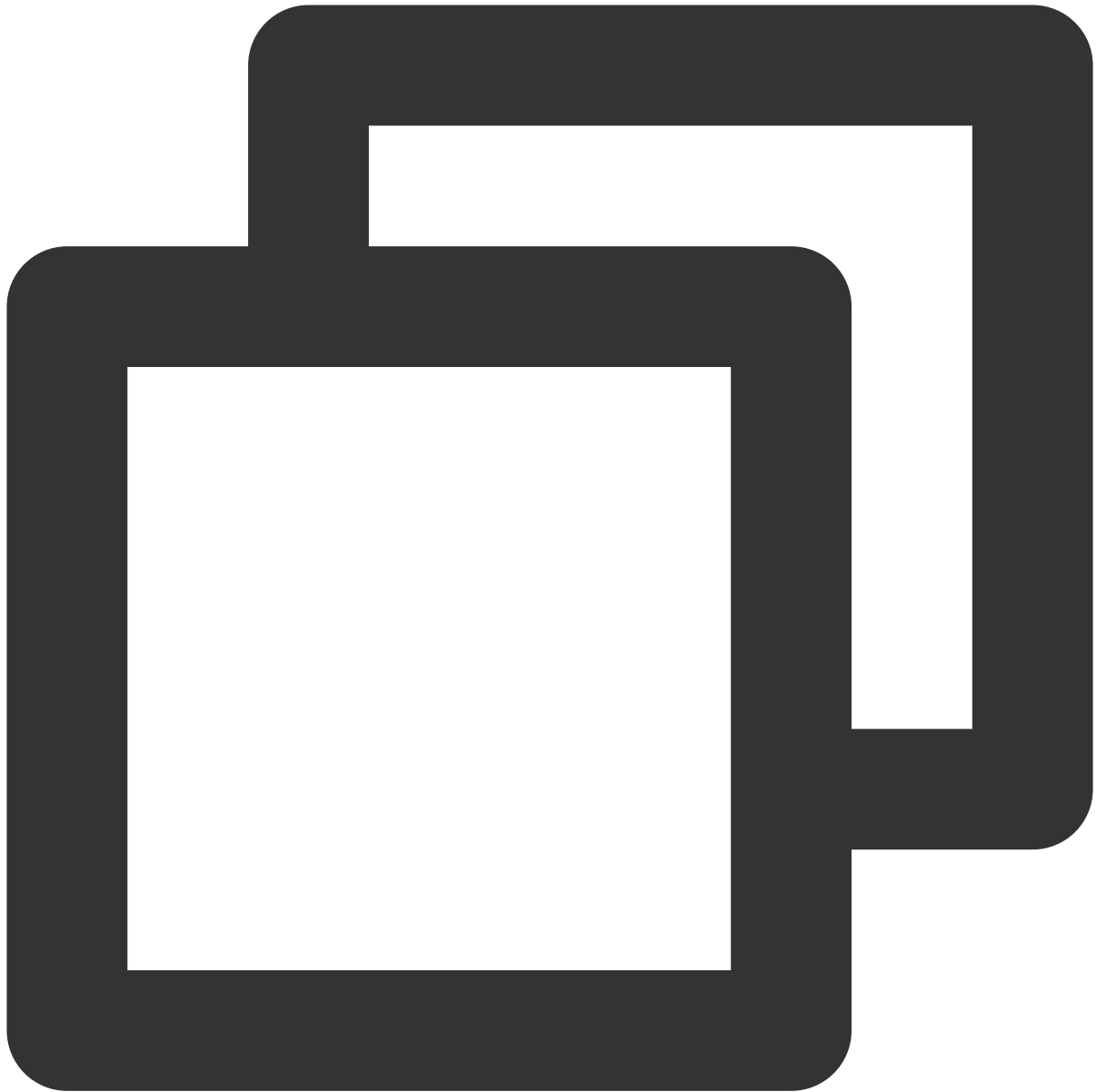




```
import torch_tccl

#args.dist_backend = "nccl"
args.dist_backend = "tccl"
torch.distributed.init_process_group(
    backend=args.dist_backend,
    init_method=args.dist_url,
    world_size=args.world_size, rank=args.rank
)
```

3. Configure TCCL environment variables.



```
export NCCL_DEBUG=INFO
export NCCL_SOCKET_IFNAME=eth0
export NCCL_IB_GID_INDEX=3
export NCCL_IB_DISABLE=0
export NCCL_IB_HCA=mlx5_bond_0,mlx5_bond_1,mlx5_bond_2,mlx5_bond_3,mlx5_bond_4,mlx5
export NCCL_NET_GDR_LEVEL=2
export NCCL_IB_QPS_PER_CONNECTION=4
export NCCL_IB_TC=160
export NCCL_IB_TIMEOUT=22
export NCCL_PXN_DISABLE=0
export TCCL_TOPO_AFFINITY=4
```

Note:

You need to enable the network topology awareness feature through `TCCL_TOPO_AFFINITY=4`.

4. Verify Pytorch.

When executing distributed training, the following prompts indicate that the communication backend has been loaded correctly.

```
vm-3-17-centos:35915:35915 [0] NCCL INFO NET/Plugin : No plugin found (libnccl-net.so), using internal implementa
vm-3-17-centos:35915:35915 [0] NCCL INFO NET/IB : Using [0]mlx5_bond_0:1/RoCE [R0]; 00B eth0:10.100.3.17<0>
vm-3-17-centos:35915:35915 [0] NCCL INFO Using network IB
NCCL version 2.12.12_TCCL_v1.5+cuda11.6
vm-3-17-centos:35919:35919 [4] NCCL INFO Bootstrap : Using eth0:10.100.3.17<0>
vm-3-17-centos:35919:35919 [4] NCCL INFO NET/Plugin : No plugin found (libnccl-net.so), using internal implementa
vm-3-17-centos:35921:35921 [6] NCCL INFO Bootstrap : Using eth0:10.100.3.17<0>
vm-3-17-centos:35920:35920 [5] NCCL INFO Bootstrap : Using eth0:10.100.3.17<0>
```

5. Software version limit

The current installation package only supports Pytorch 1.12. For other supported Pytorch and CUDA versions, please contact your pre-sales manager for support.

Note:

If running `nccl-tests` or other scenarios that require dynamically linked communication libraries, use Method 1 to install TCCL.

If you have installed NCCL, you can also use the TCCL acceleration capability through the NCCL plugins.

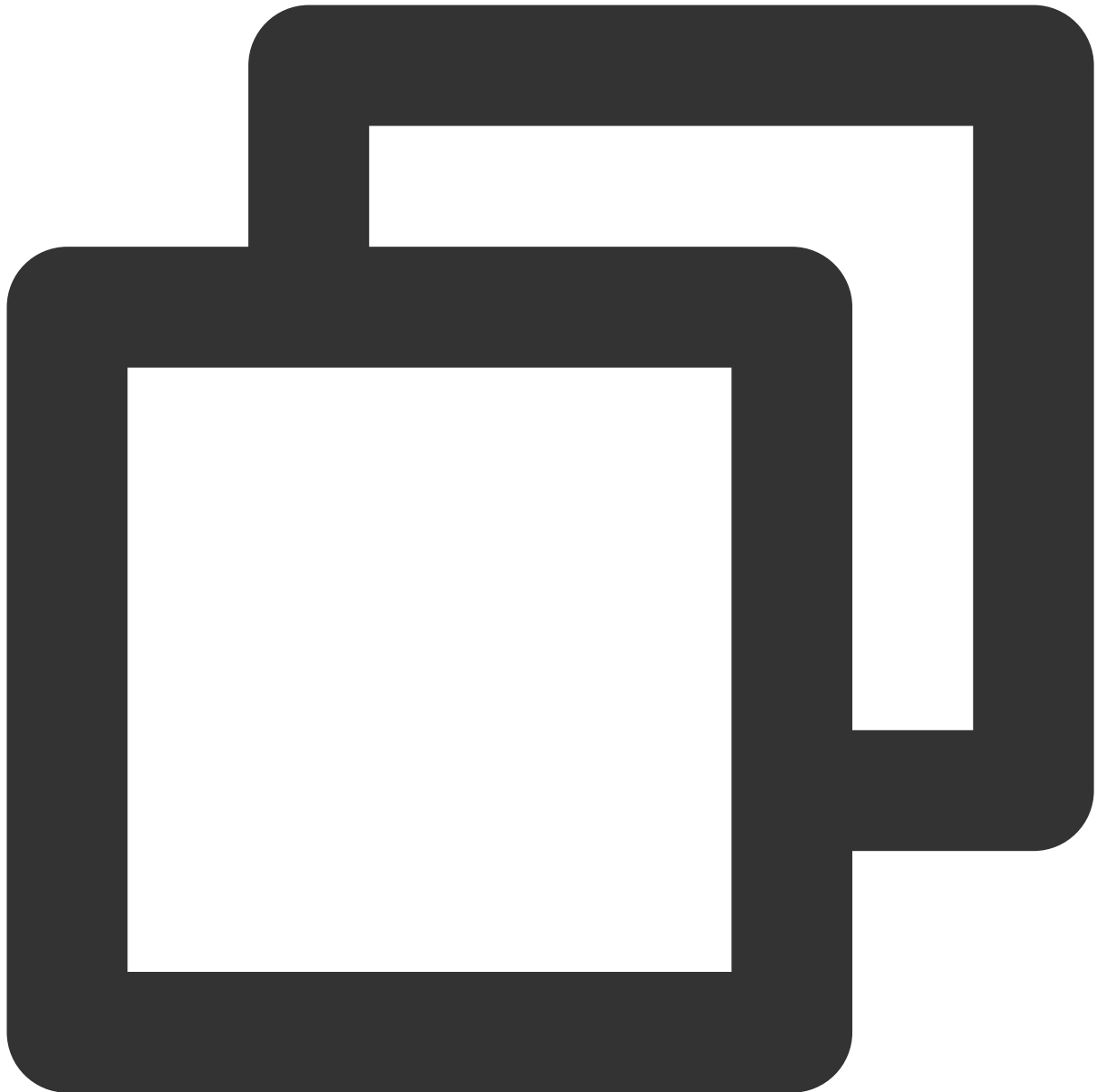
1. Install NCCL plugins.

Take Ubuntu 20.04 as an example, you can use the following commands to install plugins.



```
# Uninstall the existing tccl and nccl plugins.  
dpkg -r tccl && dpkg -r nccl-rdma-sharp-plugins  
  
# Download and install nccl 1.2 plugins.  
wget https://taco-1251783334.cos.ap-shanghai.myqcloud.com/nccl/nccl-rdma-sharp-plug  
  
# Please ensure that the version of nccl plugins used within the cluster is consist  
# wget https://taco-1251783334.cos.ap-shanghai.myqcloud.com/nccl/nccl-rdma-sharp-pl
```

If you use CentOS or TencentOS, see the following steps for installation:



```
# Uninstall the existing nccl plugins.  
rpm -e nccl-rdma-sharp-plugins-1.0-1.x86_64  
  
# Download and install nccl 1.2 plugins.  
wget https://taco-1251783334.cos.ap-shanghai.myqcloud.com/nccl/nccl-rdma-sharp-plug  
  
# Ensure that the version of nccl plugins used within the cluster is consistent. Th  
# wget https://taco-1251783334.cos.ap-shanghai.myqcloud.com/nccl/nccl-rdma-sharp-pl
```

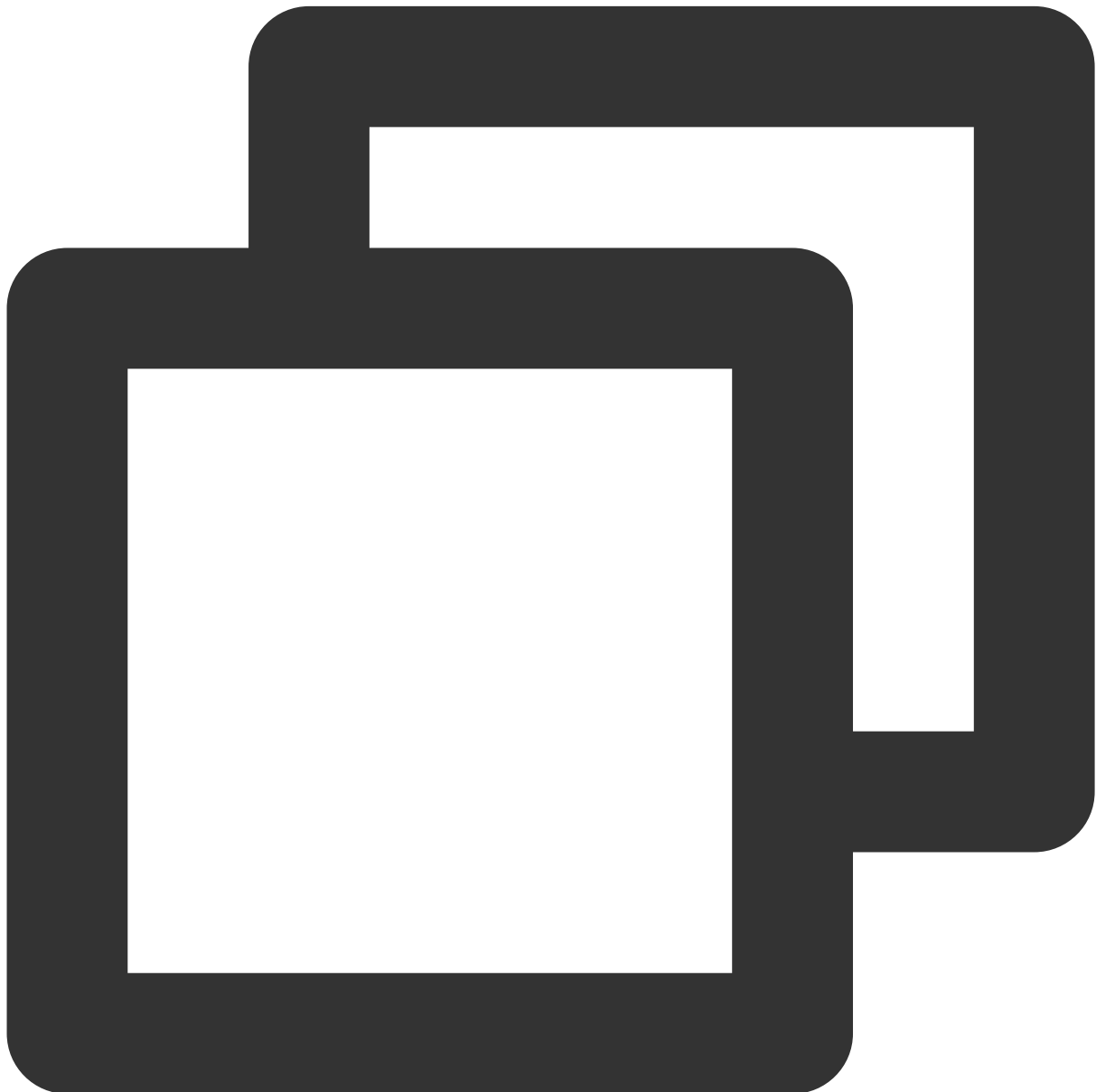
2. Obtain the topologically sorted IP list.

The NCCL plugins do not require dependency files to provide two optimizations: dynamic aggregation on bonding interfaces and global hash routing. If affinity awareness of the network topology is needed, users can achieve it through the sorted IP list.

IP sorting can be completed as follows:

Prepare the IP list file.

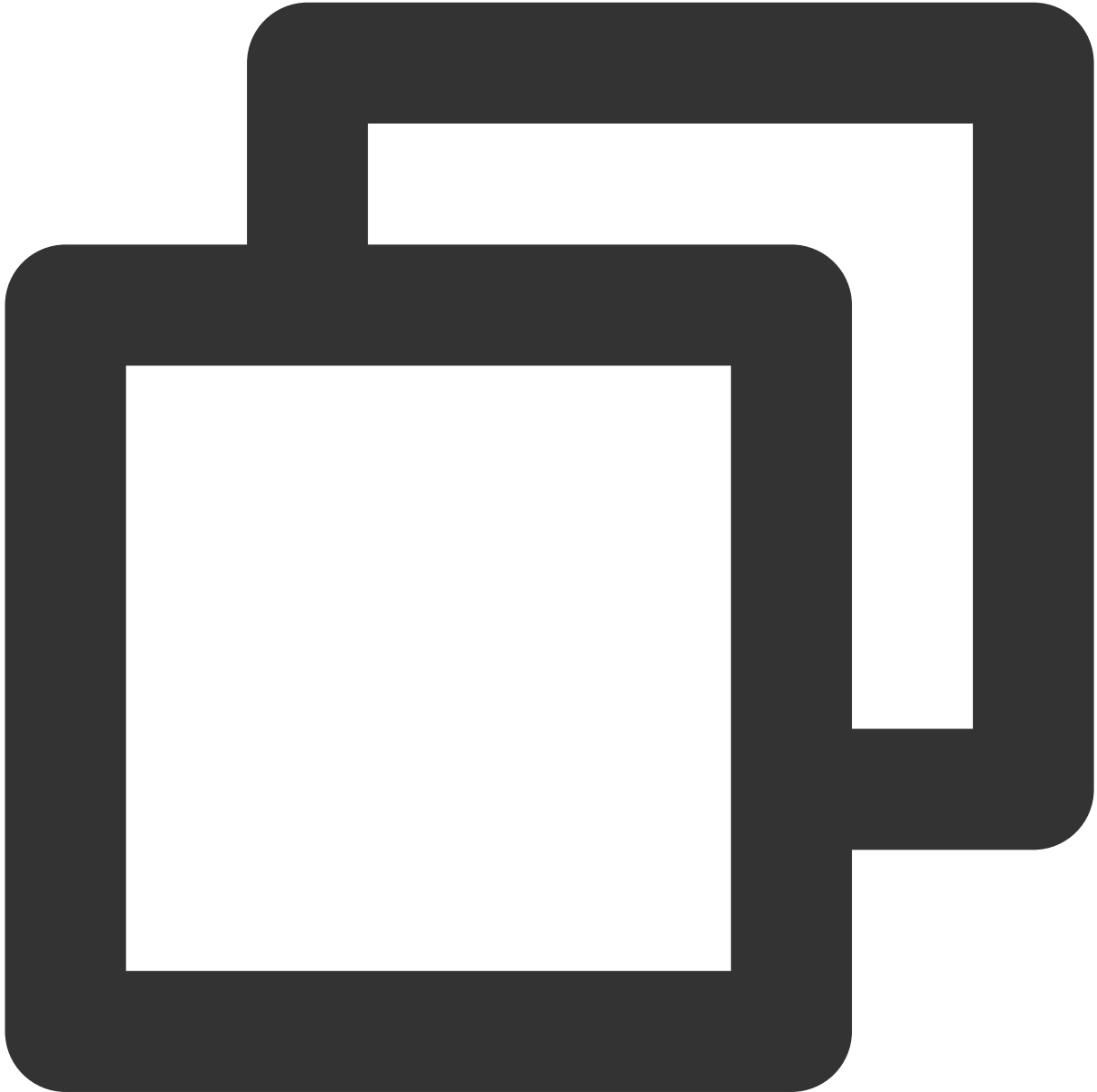
The VPC IP address can be obtained through `ifconfig eth0`, and each row has one node IP. The format is as follows:



```
root@VM-125-10-tencentos:/workspace# cat ip_eth0.txt
172.16.177.28
```

```
172.16.176.11
172.16.177.25
172.16.177.12
```

Execution sorting



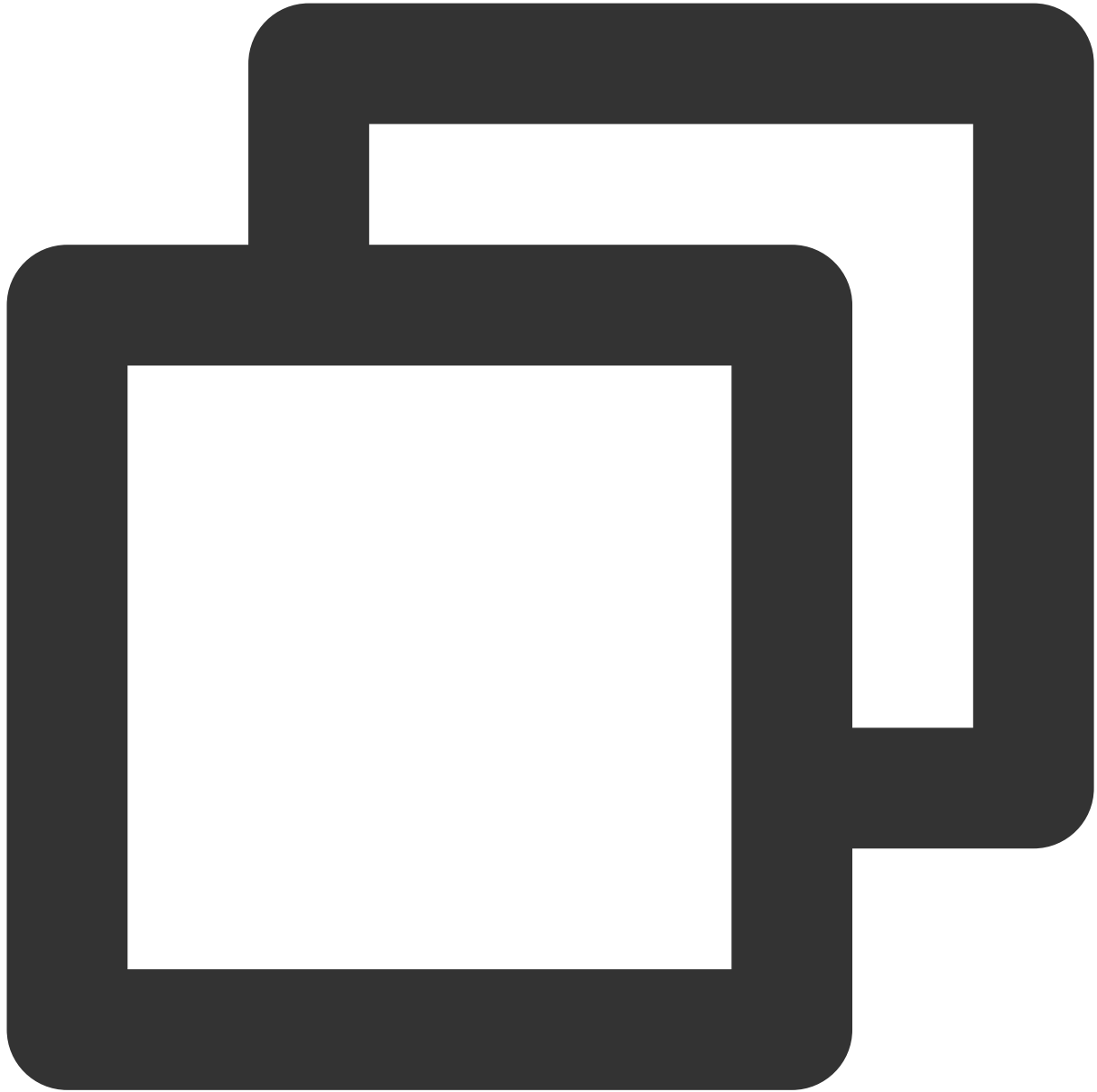
```
wget https://taco-1251783334.cos.ap-shanghai.myqcloud.com/tccl/get_rdma_order_by_ip
```

Note:

The curl tool is installed on all nodes (for Ubuntu, it can be installed via `apt install curl`).

The node executing the script can access all other nodes without ssh password.

View the sorted IP list file.



```
root@VM-125-10-tencentos:/workspace# cat hostfile.txt
172.16.176.11
172.16.177.12
172.16.177.25
172.16.177.28
```

3. Configure TCCL environment variables.

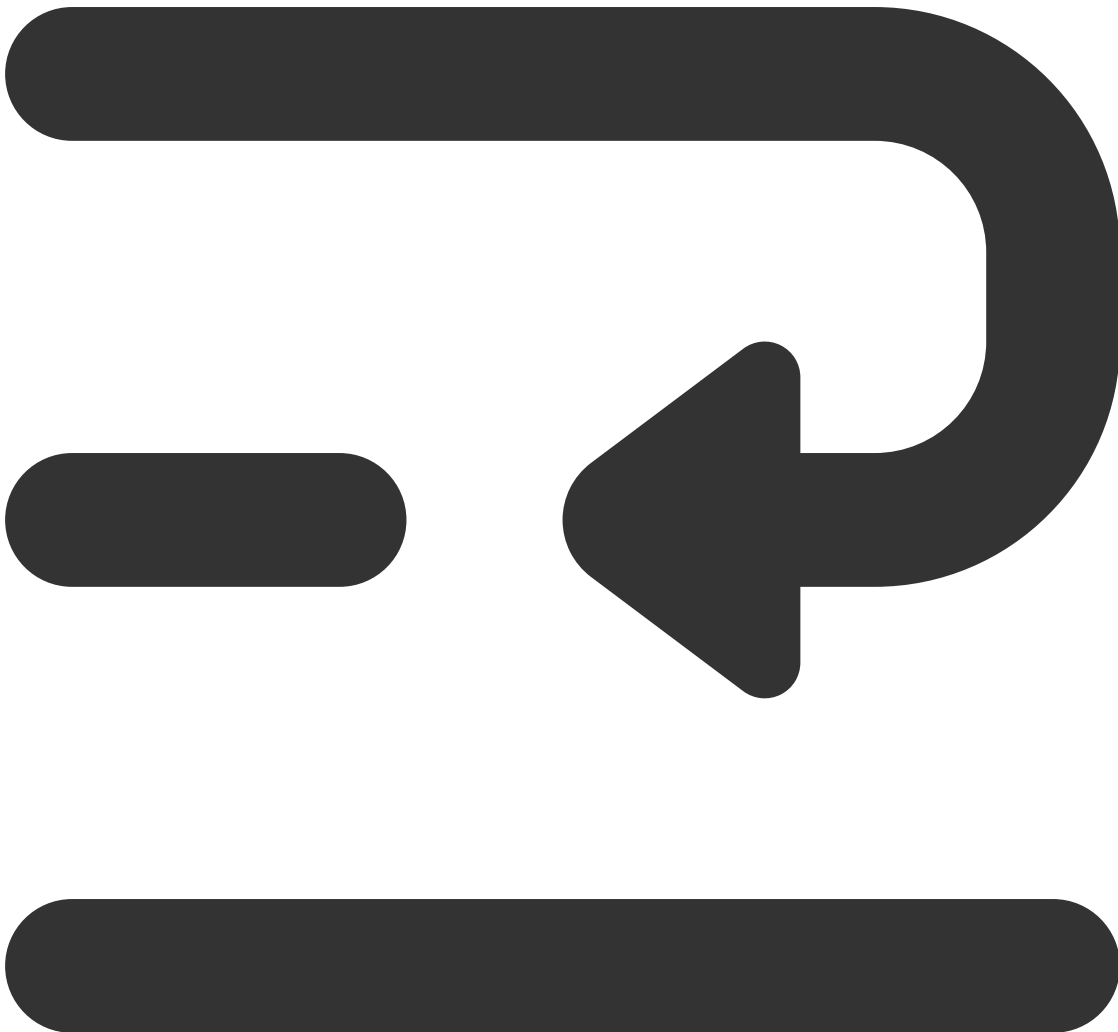


```
export NCCL_DEBUG=INFO
export NCCL_SOCKET_IFNAME=eth0
export NCCL_IB_GID_INDEX=3
export NCCL_IB_DISABLE=0
export NCCL_IB_HCA=mlx5_bond_0,mlx5_bond_1,mlx5_bond_2,mlx5_bond_3,mlx5_bond_4,mlx5
export NCCL_NET_GDR_LEVEL=2
export NCCL_IB_QPS_PER_CONNECTION=4
export NCCL_IB_TC=160
export NCCL_IB_TIMEOUT=22
export NCCL_PXN_DISABLE=0
```

```
# After the machine IP is manually sorted, there is no need to add the following va  
# export TCCL_TOPO_AFFINITY=4
```

4. Modify the startup script.

You need to modify the startup script when starting distributed training. For example, if you use the deepspeed launcher to start the training process, you need to obtain the sorted IP list, write the corresponding IP list into the hostfile, and then start the training process.

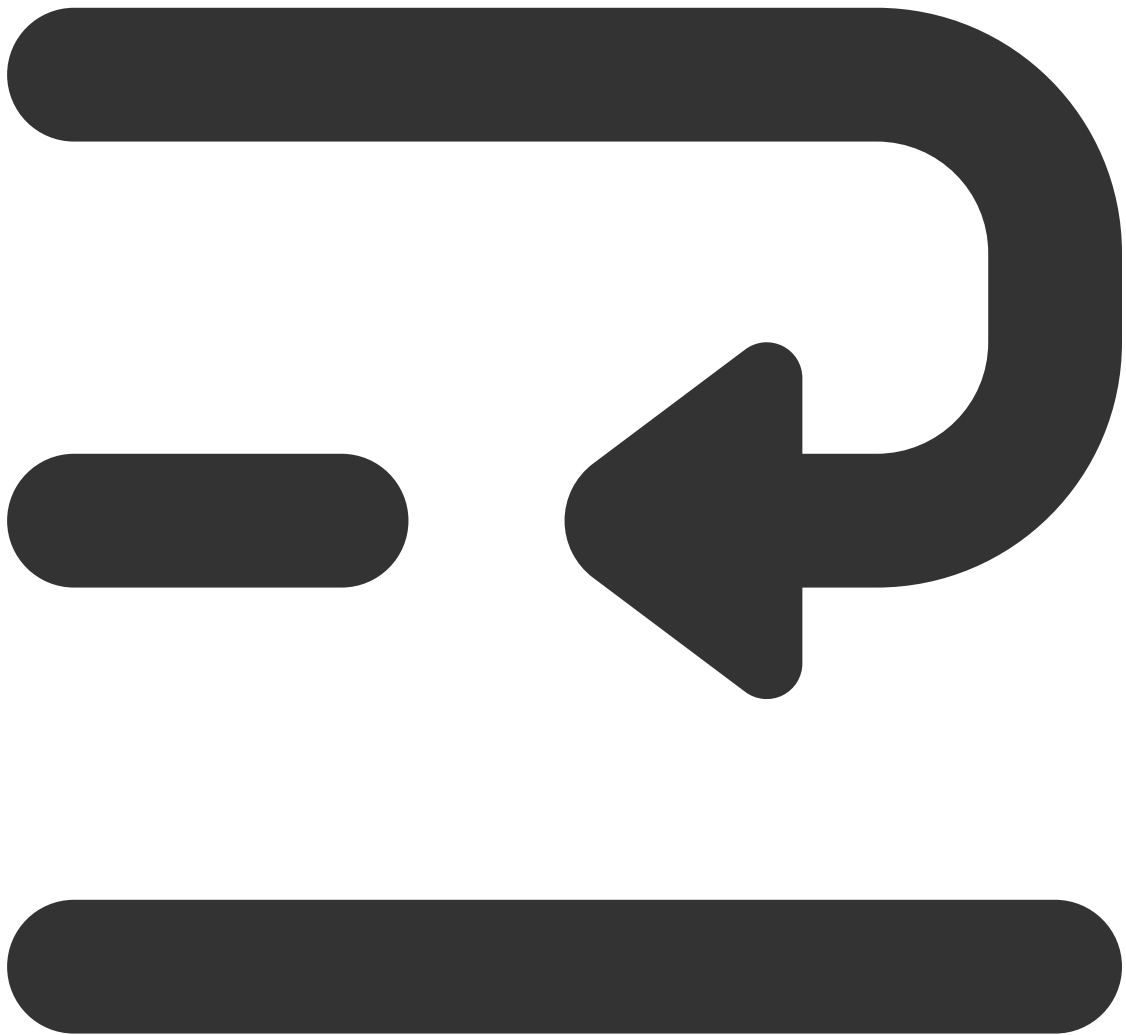


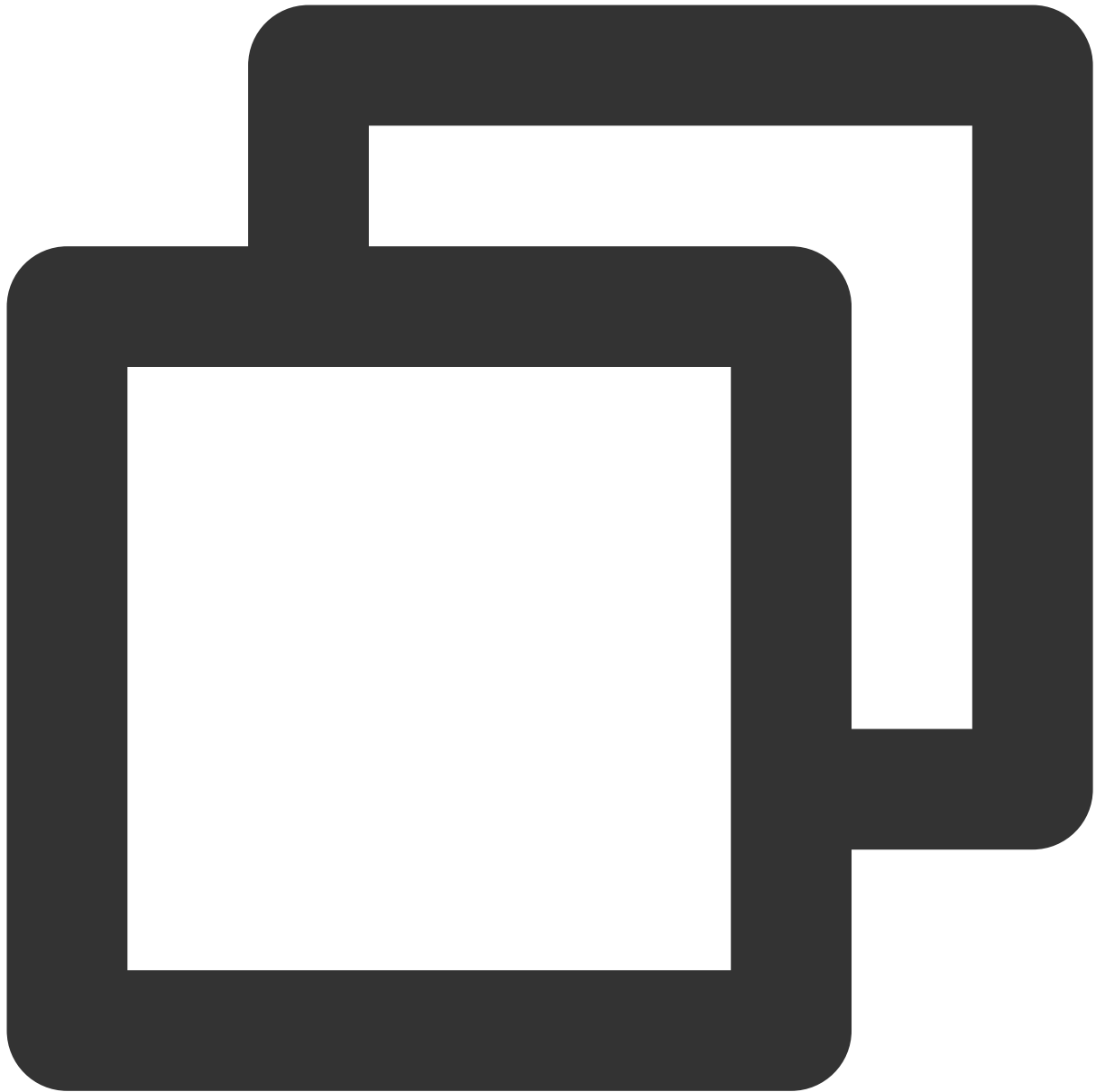


```
root@vm-3-17-centos:/workspace/ptm/gpt# cat hostfile
172.16.176.11 slots=8
172.16.177.12 slots=8
172.16.177.25 slots=8
172.16.177.28 slots=8

deepspeed --hostfile ./hostfile --master_addr 172.16.176.11 train.py
```

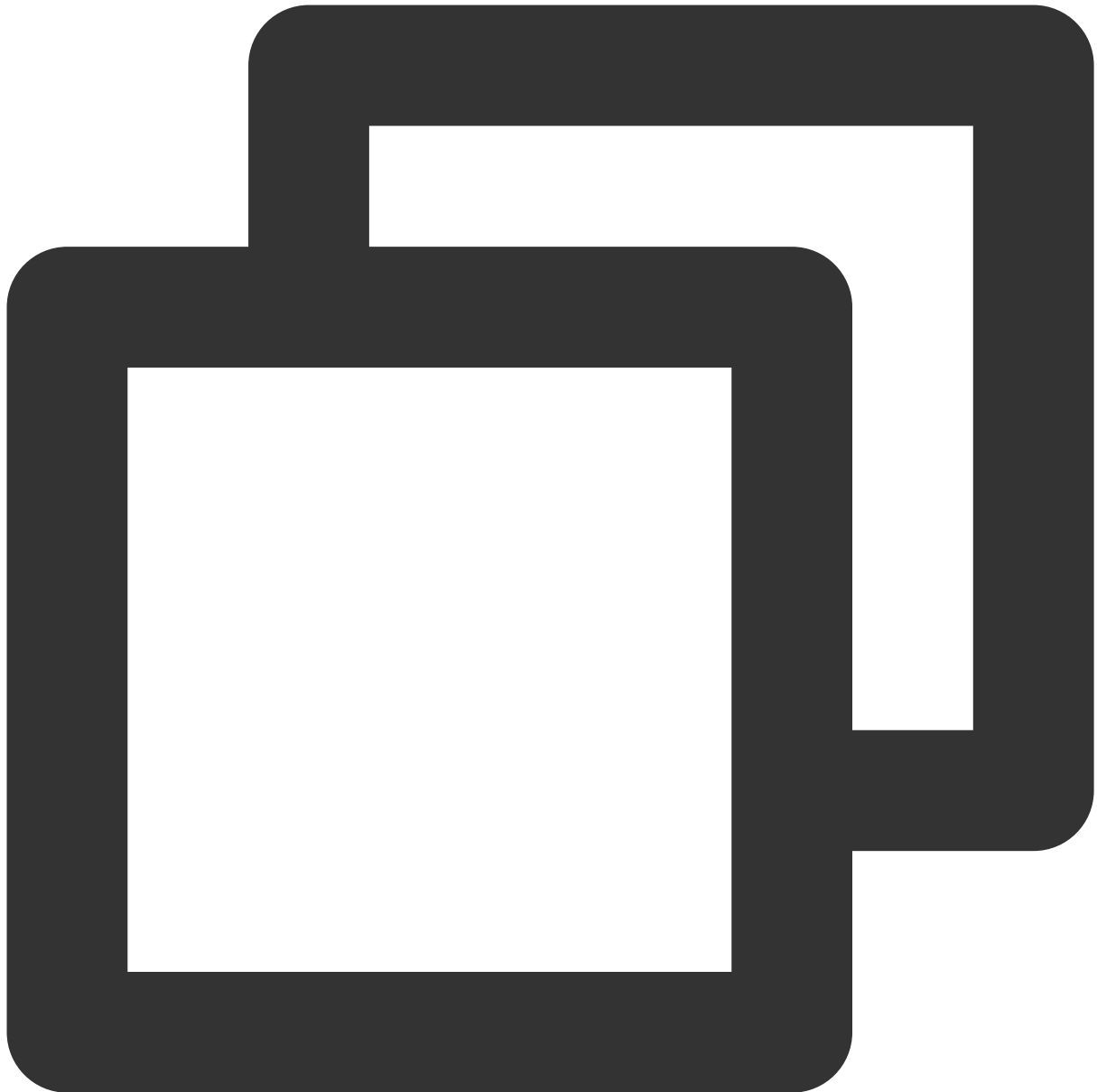
If torchrun is used to start the training process, specify the corresponding node sequence through `--node_rank` ,





```
// on 172.16.176.11
torchrun --nnodes=4 --nproc_per_node=8 --node_rank=0 --master_addr=172.16.176.11 tr
// on 172.16.176.12
torchrun --nnodes=4 --nproc_per_node=8 --node_rank=1 --master_addr=172.16.176.11 tr
// on 172.16.176.25
torchrun --nnodes=4 --nproc_per_node=8 --node_rank=2 --master_addr=172.16.176.11 tr
// on 172.16.176.28
torchrun --nnodes=4 --nproc_per_node=8 --node_rank=3 --master_addr=172.16.176.11 tr
```

If `mpirun` is used to start the training process, just sort the IP addresses in order.



```
mpirun \  
-np 64 \  
-H 172.16.176.11:8,172.16.177.12:8,172.16.177.25:8,172.16.177.28:8 \  
--allow-run-as-root \  
-bind-to none -map-by slot \  
-x NCCL_DEBUG=INFO \  
-x NCCL_IB_GID_INDEX=3 \  
-x NCCL_IB_DISABLE=0 \  
-x NCCL_SOCKET_IFNAME=eth0 \  
-x NCCL_IB_HCA=mlx5_bond_0,mlx5_bond_1,mlx5_bond_2,mlx5_bond_3,mlx5_bond_4,mlx5_bon \  
-x NCCL_NET_GDR_LEVEL=2 \  

```

```
-x NCCL_IB_QPS_PER_CONNECTION=4 \\
-x NCCL_IB_TC=160 \\
-x NCCL_IB_TIMEOUT=22 \\
-x NCCL_PXN_DISABLE=0 \\
-x LD_LIBRARY_PATH -x PATH \\
-mca coll_hcoll_enable 0 \\
-mca pml ob1 \\
-mca btl_tcp_if_include eth0 \\
-mca btl ^openib \\
all_reduce_perf -b 1G -e 1G -n 1000 -g 1
```

Installing RDMA Millisecond-Level Monitoring Component on GPU Instances

Last updated : 2024-08-20 17:08:22

Feature Introduction

Hyper Computing Cluster has the capability to achieve millisecond-level monitoring in an RDMA network environment, enabling you to monitor and analyze instantaneous network data in real-time, helping you deeply analyze network traffic modes, optimize network and improve performance, and provide strong support for your business.

Overview

This document describes how to install the millisecond-level monitoring component in the Tencent Cloud Hyper Computing Cluster environment to achieve millisecond-level performance monitoring in Tencent Cloud RDMA environments. Tencent Cloud provides two ways to view monitoring data. You can view the millisecond-level monitoring statistics on Cloud Product Monitoring or view the saved monitoring logs locally on the instance.

Note:

The startup of RDMA millisecond-level monitoring will occupy less than 0.05 CPU resources. You can decide whether to use it based on business needs.

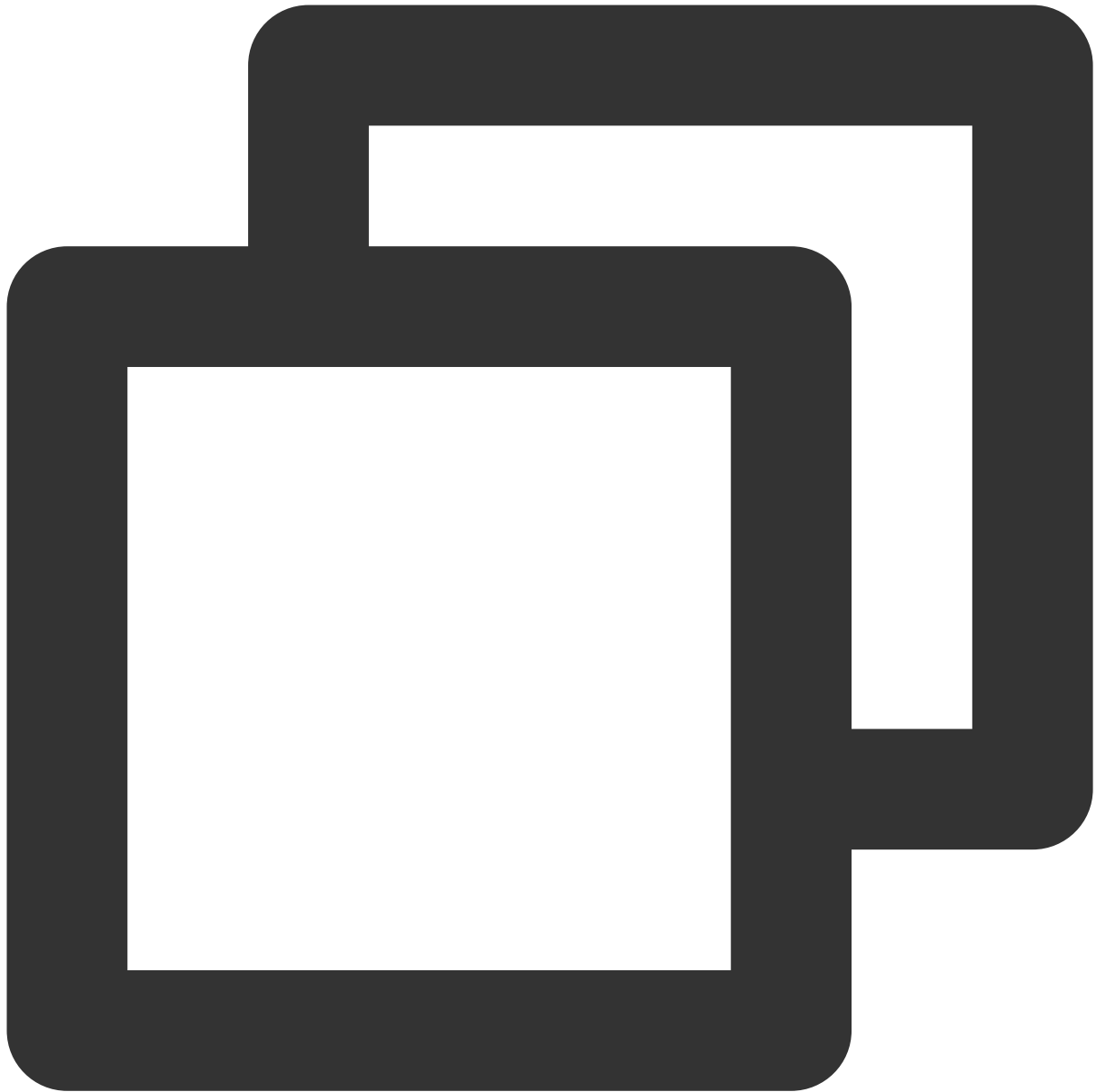
Directions

Environment Preparations

1. Create a GPU Hyper Computing ClusterPNV4sne, GPU Hyper Computing ClusterPNV4sn, or GPU Hyper Computing ClusterPNV5v Hyper Computing Cluster instance. It is recommended to select the TencentOS Server 2.4 (TK4) image.
2. Install the GPU driver and [nvidia-fabricmanager service](#) for GPU instances.

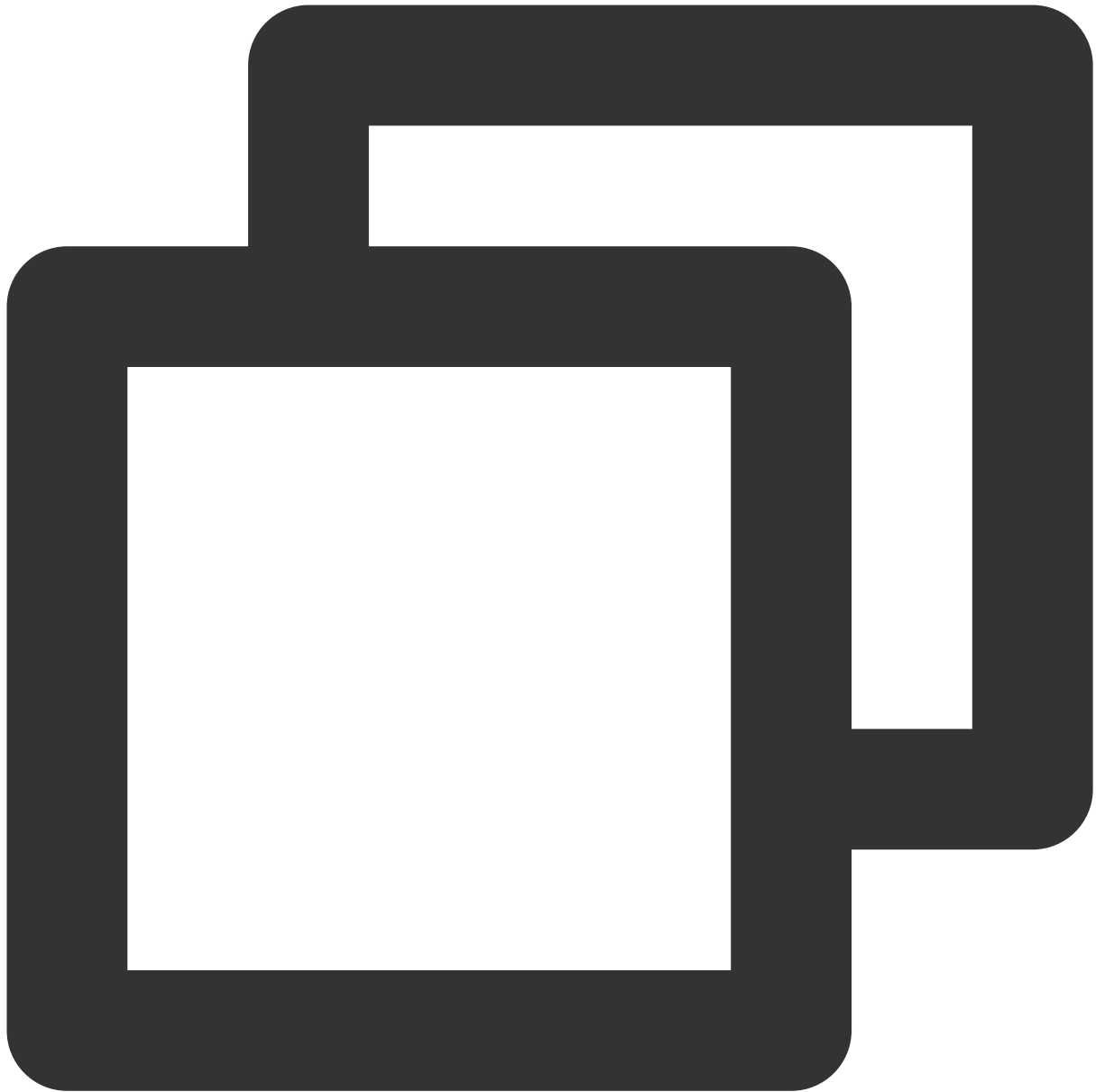
Installation Verification

1. In the TencentOS Server 2.4 (TK4) environment, you can use the following commands to install:



```
# Uninstall the existing enhanced monitoring software package
rpm -e rdma_monitor-1.0-1.tl2.x86_64
# Download and install the millisecond-level monitoring component.
# Once the software package is installed, a system service will be automatically re
wget http://mirrors.tencentyun.com/install/GPU/rdma_monitor-1.0-1.tl2.x86_64.rpm &&
```

2. Use the following command to verify if the installation is successful:



```
ps -aux | grep monitor_server
```

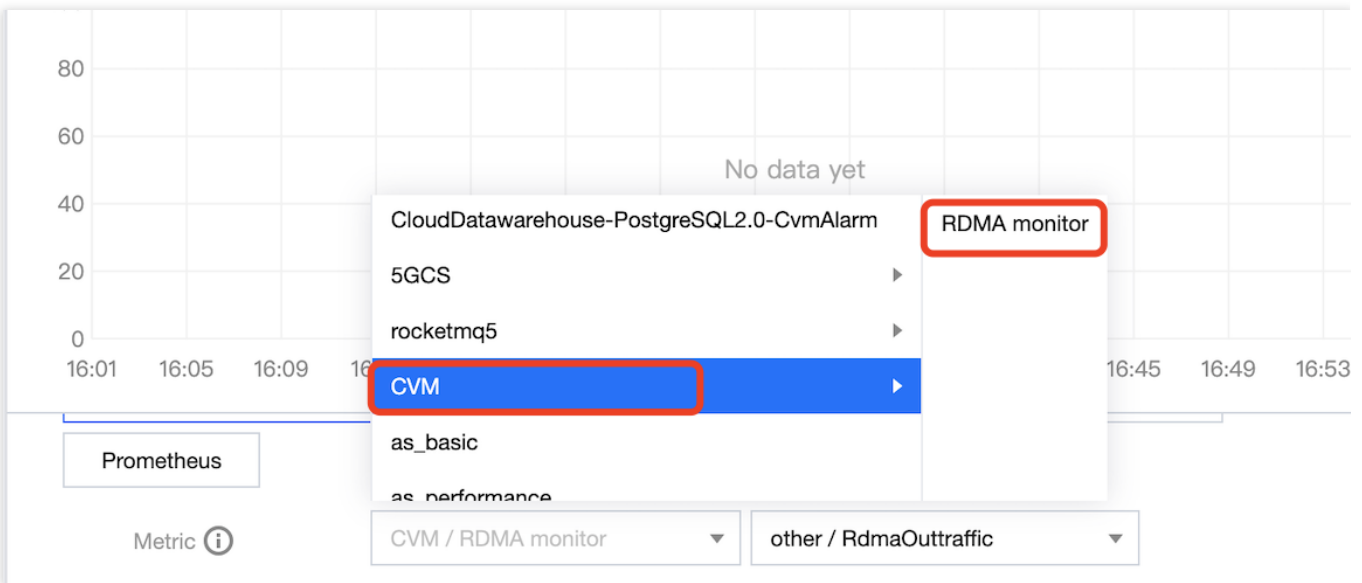
Execute the command. If the fields are displayed in red, it means the enhanced monitoring has been successfully installed and started.

```
[root@VM-16-2-tencentos ~]# ps -aux | grep monitor_server
root      3719  0.0  0.0 112824  2260 pts/5    S+   15:26   0:00 grep
root     230127  1.8  0.0 1587160 25080 pts/0    Sl+  15:20   0:06 moni
```

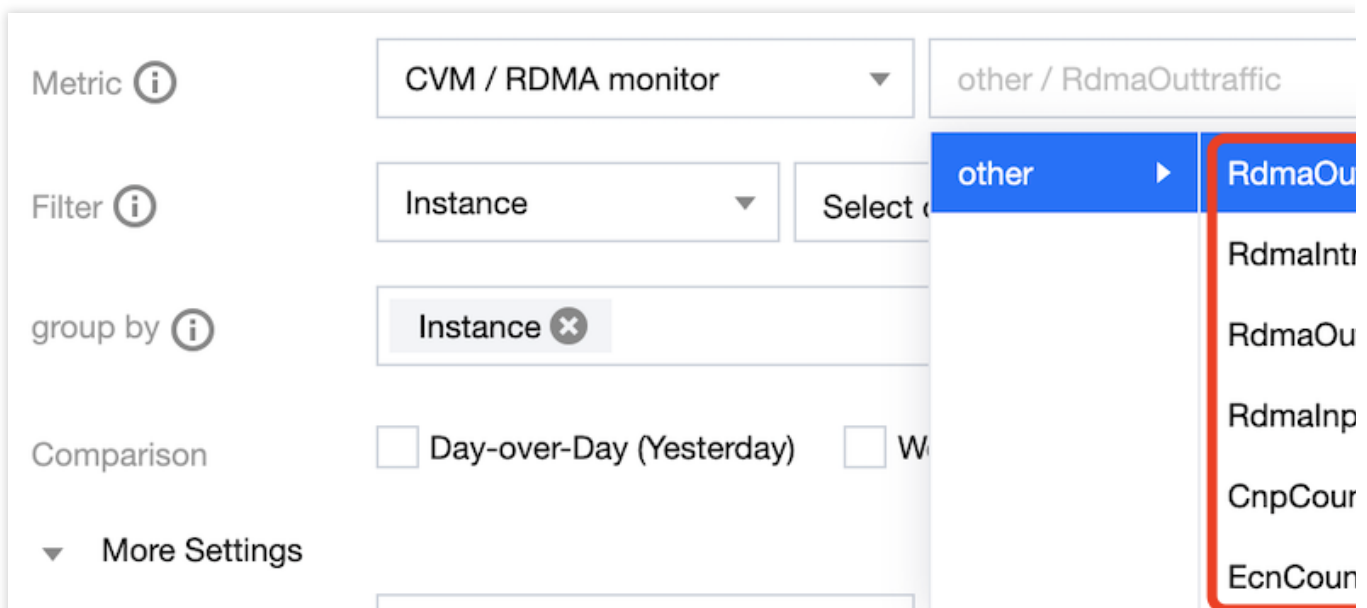
Viewing Cloud Product Monitoring Data

The data of RDMA millisecond-level monitoring can be viewed in the Cloud Product Monitoring. You can configure the required monitoring metrics in the Cloud Product Monitoring - dashboard. The directions are as follows:

1. [Create dashboard](#), and select **CVM - RDMA Monitoring** as the metric:



2. Select the RDMA millisecond-level statistical metrics you need to monitor.



Cloud Product Monitoring supports viewing the following statistical data. You can configure it in the Cloud Product Monitoring dashboard as needed.

--	--	--	--	--	--

English Metric Name	Chinese Metric Name	Metric Description (optional)	Unit	Dimension	Statistical Granularity
RxHpbwAvg	Millisecond-level_average of RDMA network interface received bandwidth	The millisecond-level statistical granularity average of the RDMA network interface received bandwidth within 10 seconds	Mbps	InstanceId	10s, 60s, 300s, 3,600s
RxHpbwMax	Millisecond-level_maximum value of RDMA network interface received bandwidth	The millisecond-level statistical granularity maximum value of the RDMA network interface received bandwidth within 10 seconds	Mbps	InstanceId	10s, 60s, 300s, 3,600s
RxHpbwMin	Millisecond-level_minimum value of RDMA network interface received bandwidth	The millisecond-level statistical granularity minimum value of the RDMA network interface received bandwidth within 10 seconds	Mbps	InstanceId	10s, 60s, 300s, 3,600s
RxHpbwP50	Millisecond-level_the 50th percentile of RDMA network interface received bandwidth	The millisecond-level statistical granularity 50th percentile of the RDMA network interface received	Mbps	InstanceId	10s, 60s, 300s, 3,600s, 86,400s

		bandwidth from lowest to highest within 10 seconds			
RxHpbwP90	Millisecond-level_the 90th percentile of RDMA network interface received bandwidth	The millisecond-level statistical granularity 90th percentile of the RDMA network interface received bandwidth from lowest to highest within 10 seconds	Mbps	InstanceId	10s, 60s, 300s, 3,600s
TxHpbwAvg	Millisecond-level_average of RDMA network interface transmitted bandwidth	The millisecond-level statistical granularity average of the RDMA network interface transmitted bandwidth within 10 seconds	Mbps	InstanceId	10s, 60s, 300s, 3,600s
TxHpbwMax	Millisecond-level_maximum value of RDMA network interface transmitted bandwidth	The millisecond-level statistical granularity maximum value of the RDMA network interface transmitted bandwidth within 10 seconds	Mbps	InstanceId	10s, 60s, 300s, 3,600s
TxHpbwMin	Millisecond-level_minimum value of RDMA network interface transmitted bandwidth	The millisecond-level statistical granularity minimum value of the RDMA network	Mbps	InstanceId	10s, 60s, 300s, 3,600s

		interface transmitted bandwidth within 10 seconds			
TxHpbwP50	Millisecond-level_the 50th percentile of RDMA network interface transmitted bandwidth	The millisecond-level statistical granularity 50th percentile of the RDMA network interface transmitted bandwidth from lowest to highest within 10 seconds	Mbps	InstanceId	10s, 60s, 300s, 3,600s
TxHpbwP90	Millisecond-level_the 90th percentile of RDMA network interface transmitted bandwidth	The millisecond-level statistical granularity 90th percentile of the RDMA network interface transmitted bandwidth from lowest to highest within 10 seconds	Mbps	InstanceId	10s, 60s, 300s, 3,600s

3. Select the Hyper Computing Cluster instance ID you need to monitor.

Filter i

Instance ▼

Select object ▼

4. click **OK** to quickly create Dashboard.

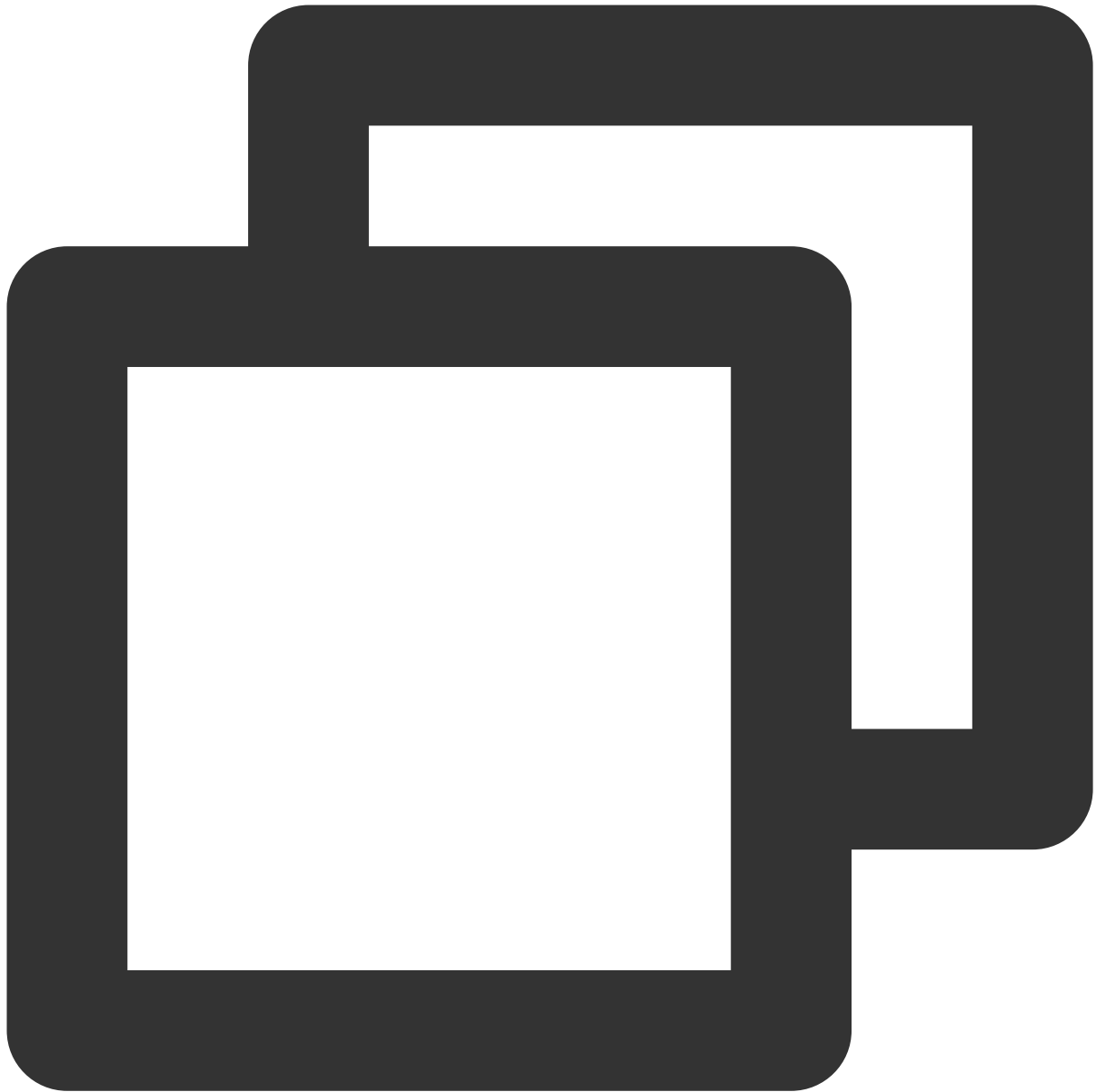
New Dashboard ✕

Dashboard Name

Folder

Viewing Local Monitoring Data

RDMA millisecond-level monitoring can view bandwidth data monitoring with a minimum granularity of 10ms, but Cloud Product Monitoring only supports data reporting with a minimum granularity of 10s. If users want to obtain more precise network interface monitoring data, they can use the following command to save millisecond-level data for local viewing.



```
# monitor_client With the enhanced monitoring automatically installed, /tmp/monitor
monitor_client -r -p raw > /tmp/monitor.log
# -r continuously obtain data from the last 10s
# -p; print selection
# -p summary; default value; print statistical information
# -p raw; print original data points
# -p all; print both statistical information and original data points
# You can use monitor_client -h to view more parameter descriptions.
```

To view the recorded monitoring data, you can analyze the monitoring records as needed. The format of the monitoring records is as follows:

```
Device: bond3, Transmitted data points: 1000, Timestamp: 1697616573
Data Point 0: 0
Data Point 1: 0
Data Point 2: 0
Data Point 3: 0
Data Point 4: 0
Data Point 5: 0
Data Point 6: 0
Data Point 7: 0
Data Point 8: 0
Data Point 9: 0
Data Point 10: 0
Data Point 11: 0
Data Point 12: 0
Data Point 13: 0
Data Point 14: 0
Data Point 15: 0
Data Point 16: 0
```

Note:

The meanings of some parameters in the figure are explained as follows:

Device: RDMA network interface name.

Received data points: the number of data points collected on the receiving side within 10s. There are 1,000 points collected within 10s, which means one data point was collected every 10ms and each data point represents the corresponding 10ms received bandwidth.

Timestamp: timestamp during collection

Data Point n: the received bandwidth collected after the timestamp $n \times 10$ ms. Each data point is spaced 10 ms from the preceding and following points.