ALIBABA CLOUD

Alibaba Cloud

Machine Learning Platform for AI

DLC-Cloud-native Deep Learning Model Training

Document Version: 20210520

Alibaba Cloud

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.

2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company or individual in any form or by any means without the prior written consent of Alibaba Cloud.

3. The content of this document may be changed because of product version upgrade, adjustment, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and an updated version of this document will be released through Alibaba Cloud-authorized channels from time to time. You should pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.

4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides this document based on the "status quo", "being defective", and "existing functions" of its products and services. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not take legal responsibility for any errors or lost profits incurred by any organization, company, or individual arising from download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, take responsibility for any indirect, consequential, punitive, contingent, special, or punitive damages, including lost profits arising from the use or trust in this document (even if Alibaba Cloud has been notified of the possibility of such a loss).

5. By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.

6. Please directly contact Alibaba Cloud for any errors of this document.

Document conventions

Style	Description	Example
Danger	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	**Danger:** Resetting will result in the loss of user configuration data.
Warning	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	**Warning:** Restarting will cause business interruption. About 10 minutes are required to restart an instance.
Notice	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	**Notice:** If the weight is set to 0, the server no longer receives new requests.
Note	A note indicates supplemental instructions, best practices, tips, and other content.	**Note:** You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click **Settings**> **Network**> **Set network type**.
Bold	Bold formatting is used for buttons , menus, page names, and other UI elements.	Click **OK**.
`Courier font`	Courier font is used for commands	Run the `cd /d C:/window` command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	This format is used for an optional value, where only one item can be selected.	`ipconfig [-all|-t]`
{} or {a|b}	This format is used for a required value, where only one item can be selected.	`switch {active|stand}`

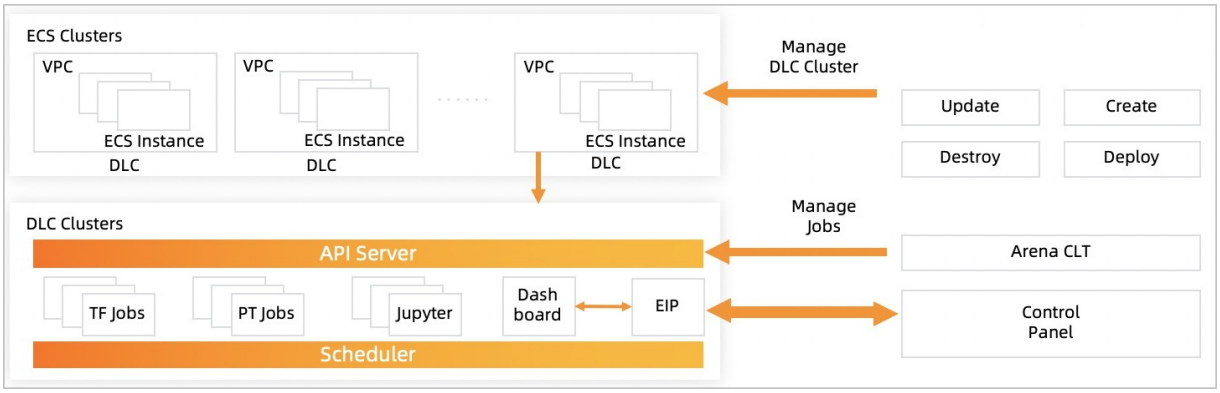
Table of Contents

1.Overview 05
2.Authorization 06
3.Prerequisites 07
3.1. Install the public-facing SLB ingress controller 07
3.2. Mount volumes to ACK clusters 08
3.2.1. Mount a NAS file system to an ACK cluster 08
3.2.2. Mount an OSS bucket to an ACK cluster 10
3.2.3. Mount a CPFS to an ACK cluster 13
4.Manage clusters 15
4.1. Overview 15
4.2. Add an ACK cluster 16
5.Manage deep learning jobs 18
5.1. Images 18
5.2. Manage jobs in DLC Dashboard 19
5.3. Manage jobs in Arena 21

1.Overview

Deep Learning Containers (DLC) of Machine Learning Platform for AI is a deep learning platform developed based on Alibaba Cloud Container Service for Kubernetes (ACK). It provides stable, easy-to-use, scalable, and high-performance runtimes for training deep learning models.

DLC architecture



DLC integrates deep learning frames and network optimization technologies of Machine Learning Platform for AI. DLC offers distributed computing capabilities that support near-linear expansion. The NVIDIA P128 GPU can boost the speedup of parallel computing to higher than 100 times. In scenarios where tens of millions of image datasets or hundreds of thousands of categories need to be processed, the performance of model training by DLC is over eight times higher than open source deep learning frameworks. For search, recommendation, advertising, and media feeds services in the Internet industry, DLC is capable of handling hundreds of billions of samples and tens of billions of features. DLC allows you to train models on thousands of nodes in parallel. In these scenarios, the performance of model training by DLC is over five times higher than open source deep learning frameworks.

DLC supports the following features:

- Provides a distributed deployment solution for conducting data parallelism, model parallelism, and hybrid parallelism.
- Allows you to use existing ACK clusters to train deep learning models.
- Supports open source Kubernetes interfaces. This allows you to submit training jobs with user-provided images.
- Provides the training job management platform DLC Dashboard, which is deployed in ACK clusters. It allows you to submit jobs and monitor the progress of each job in a visualized manner.
- Allows you to use Arena and Kubectl to submit, manage, and view jobs. Arena is a Kubernetes-based command-line tool empowered by Artificial Intelligence (AI). Kubectl is a command-line tool for Kubernetes clusters.
- Allows you to monitor GPU resource usage in real time to facilitate task scheduling.

2.Authorization

Before you use Deep Learning Containers (DLC), you must create the required service linked role.

Prerequisites

Alibaba Cloud Machine Learning Platform for AI is activated. For more information, see Purchase.

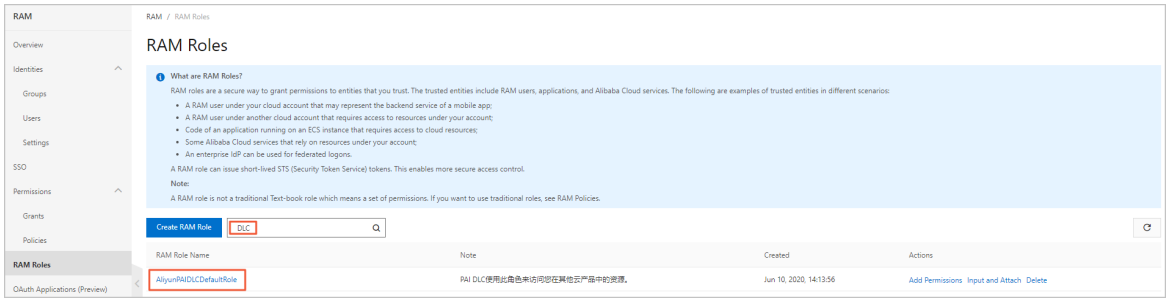
Context

To deploy services in your Alibaba Cloud Container Service for Kubernetes (ACK) cluster, DLC must first install the required components in the cluster. Therefore, you must use your Alibaba Cloud account to authorize DLC to access your ACK cluster.

> **Note** You cannot complete the authorization as a Resource Access Management (RAM) user.

Procedure

1. Navigate to the **DLC-Cloud-native Deep Learning Model Training** page.
   i. Log on to the Machine Learning Platform for AI console.
   ii. In the left-side navigation pane, choose **Model Training > DLC-Cloud-native Deep Learning Model Training**.
2. Click **Authorize Now** to open the **Cloud Resource Access Authorization** page.
3. Click **Agree to Authorization**.
4. After the RAM role is created, you can find it on the **RAM Roles** page of the RAM console. For more information, see View the basic information of a RAM role.



3.Prerequisites

3.1. Install the public-facing SLB ingress controller

You can add existing Container Service for Kubernetes (ACK) clusters to Deep Learning Containers (DLC) or create new ACK clusters on the DLC platform. If you need to use DLC Dashboard, you must first install the public-facing SLB ingress controller in your ACK cluster.

Context

For more information about how to create an ACK cluster, see Create a dedicated Kubernetes cluster.

Check whether the public-facing SLB ingress controller is installed

1. Navigate to the details page of the ACK cluster.
   i. Log on to the ACK console.
   ii. In the left-side navigation pane, click **Clusters**. Then, click the name of the cluster that you want to view.
2. On the **Cluster Information** page, click the **Cluster Resources** tab and check the configuration of **Nginx Ingress SLB**. If a hyperlink is displayed, the ingress controller is installed.
3. Check whether the **Address Type** parameter is set to **Public Network**.
   i. On the **Cluster Resources** tab of the **Cluster Information** page, click the hyperlink next to **Nginx Ingress SLB**.
   ii. In the **Server Load Balancer** console, choose **Instances > Instances**.
   iii. In the **Basic Information** section of the **Instance Details** tab, verify that the **Address Type** parameter is set to **Public Network**.

Install the public-facing SLB ingress controller when you create an ACK cluster

When you purchase an ACK cluster, select **Install Ingress Controllers** in the **Ingress** section, and set the **SLB Network Type** parameter to **Public Network**. For more information, see Create a dedicated Kubernetes cluster.

Install the public-facing SLB ingress controller in a created ACK cluster

1. Navigate to the details page of the ACK cluster.
   i. Log on to the ACK console.
   ii. In the left-side navigation pane, click **Clusters**. Then, click the name of the cluster that you want to view.
2. Log on to the ACK console. Click the name of the cluster that you want to manage to navigate to the details page of the cluster. In the left-side navigation pane, click **Add-ons**.
3. In the **System Add-ons** section, click **Install** next to **Nginx Ingress Controller**.
4. On the **Basic Information** tab of the details page, click **Rebind Domain Name** next to the

testing domain.

3.2. Mount volumes to ACK clusters

3.2.1. Mount a NAS file system to an ACK cluster

This topic describes how to mount a performance NAS file system to an Alibaba Cloud Container Service for Kubernetes (ACK) cluster. Deep Learning Containers (DLC) allows you to store source data in NAS file systems.

Step 1: Select a storage type

Multiple types of NAS file systems are provided to meet your requirements.

Parameter	Capacity	Performance	Extreme	CPFS
Latency	3 ms~10 ms	1 ms~2 ms	Hundreds of microseconds	1 ms~2 ms
IOPS	15 KB	50 KB	• Standard: 10 KB • Advanced: 30 KB	100 MB
Throughput	10 GB/s	20 GB/s	• Standard: > 150 MB/s • Advanced: > 300 MB/s	1 TB/s
Capacity	10 PB	1 PB	32 TB	100 PB
Load type	General	General	Random low I/O and container services	Paralleled reads and writes
Scenarios	Website content storage and log file backup	Content production and remote desktop sharing	Shared storage for containers of high-performance websites	High-performance computing and intelligent analytics

We recommend that you choose Cloud Paralleled File System (CPFS) in deep learning scenarios. You cannot mount a CPFS type NAS file system to an ACK cluster in the ACK console. To mount a CPFS type NAS file system to an ACK cluster, see Use CPFS volumes in ACK clusters.

Step 2: Create a NAT file system

1. For more information, see Manage file systems.
2. For more information about how to add a mount target, see Manage mount targets.When you add a mount target, set **Mount Target Type** to **VPC**, and specify **VPC default permission group (all allowed)** as **Access Group**. The **VPC network** and **VSwitch** must be the same as those of your ACK cluster.
3. Mount the NAS file system to the ACK cluster.
   i. Log on to the ACK console.
   ii. In the left-side navigation pane, choose **Clusters > Nodes**.

iii. On the **Nodes** page, click the ID of the target node in the **Name/IP Address/Instance ID** column.

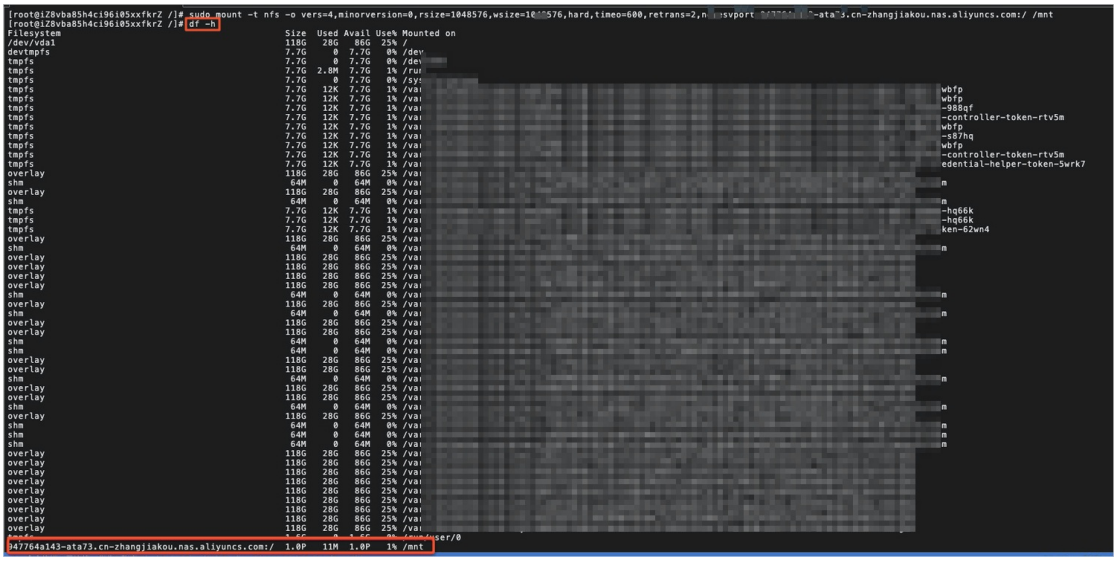
iv. On the **Instance Details** page of the Elastic Compute Service (ECS) console, click **Connect** next to **Basic Information** to connect to the ECS instance through Shell.

v. Log on to the NAS console. On the **File System List** page, click the ID of the target file system in the **File System ID/Name** column.

vi. On the **Mounting Use** page, click the icon in the **Mount Command** column to copy the **mount command**.

vii. Run the **mount command** in Shell. For more information, see Mount an NFS file system.

viii. After the file system is mounted, you can run the `df -h` command to view the mounted file system.



Step 3: Mount the NAS file system to the ACK cluster

You can mount the NAS file system to the ACK cluster in the following ways:

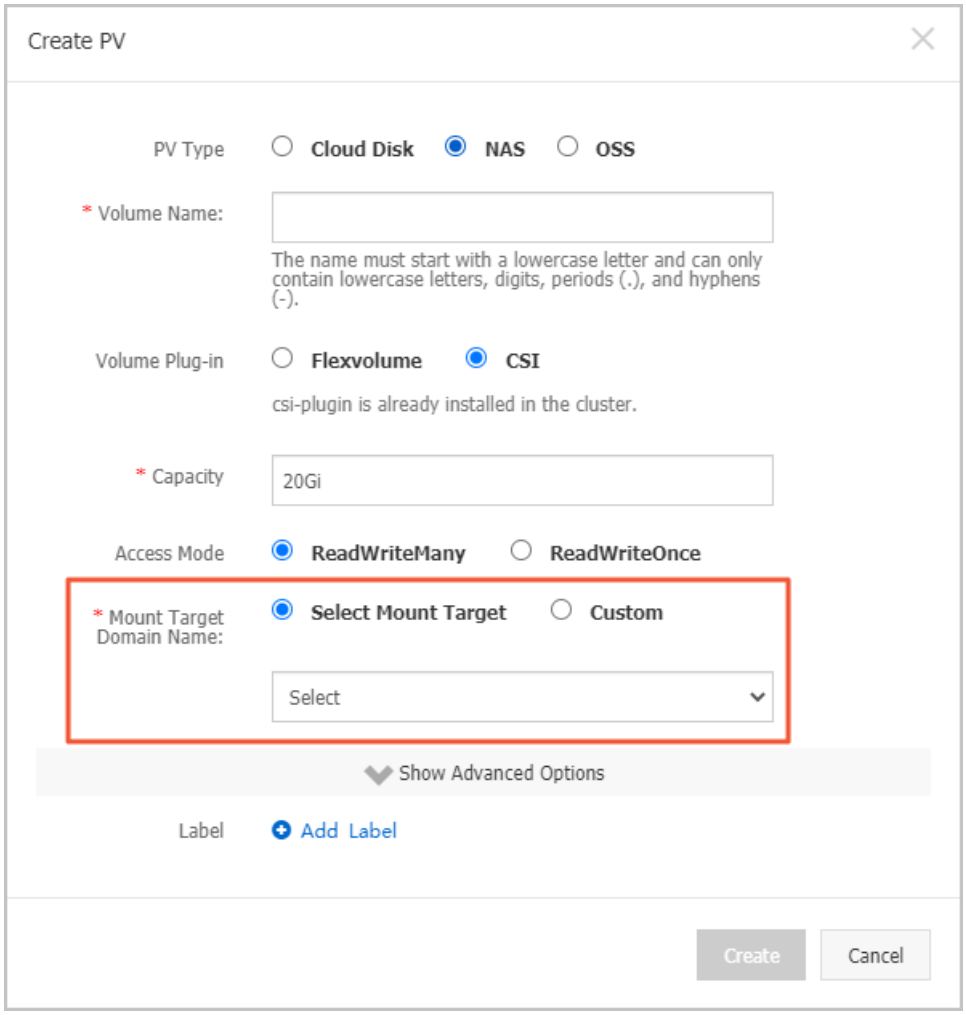
- Mount the NAS file system as a static volume
  - Directly mount the NAS file system as a volume.
  - Use a pair of **persistent volume (PV) and persistent volume claim (PVC)**. We recommend that you choose this method to mount NAS file systems to ACK clusters used in DLC.
- Mount the NAS file system as a dynamic volume

In this topic, a pair of **PV and PVC** is used as an example. For more information about how to mount a NAS file system in other ways, see Use NAS volumes.

1. Log on to the ACK console.
2. In the left-side navigation pane, choose **Clusters > Persistent Volumes**.
3. Create a PV.

i. On the **Persistent Volumes** page, select the target cluster and click **Create**.

ii. Enter a name in the **Volume Name** field, and set **Mount Target Domain Name** to **Select Mount Target**.If you are using a capacity or performance NAS file system, you can directly select a mount target from the list. If you want to mount the NAS file system to a subdirectory, click **Show Advanced Options**, and enter a path in the **Subdirectory** field.



iii. Click **Create**.

4. Create a PVC.

i. On the **Persistent Volumes** page, click the **Persistent Volume Claims** tab.

ii. Select the target cluster from the **Cluster** list and click **Create**.

iii. Click the **Select PV** hyperlink next to **Existing Volumes**, and then select the created PV.

iv. Click **Create**.

3.2.2. Mount an OSS bucket to an ACK cluster

Deep Learning Containers (DLC) allows you to store source data in Object Storage Service (OSS). This topic describes how to mount an OSS bucket to a Container Service for Kubernetes (ACK) cluster.

Prerequisites

An OSS bucket is created. For more information, see Create buckets.

> **Note** If a node and an OSS bucket are located in the same region, you can use the internal endpoint of the OSS bucket when you mount the bucket to the node.

Create a PV

1. Log on to the ACK console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Volumes > Persistent Volumes**.
5. Click the **Persistent Volumes** tab. On the **Persistent Volumes** tab, click **Create** in the upper-right corner.
6. In the **Create PV** dialog box, configure the parameters.

Parameter	Description
PV Type	You can select Cloud Disk, NAS, or OSS. In this example, select OSS.
Volume Name	The name of the persistent volume (PV). The name must be unique in the namespace. In this example, enter pv-oss.
Volume Plug-in	You can select Flexvolume or CSI. In this example, select CSI.
Capacity	The capacity of the PV.
Access Mode	Default value: ReadWriteMany.
AccessKey ID	The AccessKey pair that is required to access the OSS bucket.
AccessKey Secret	
Optional Parameters	Configure custom parameters for the OSS bucket in the format of `-o *** -o ***`.
Bucket ID	The name of the OSS bucket that you want to use. Click **Select Bucket**. In the dialog box that appears, select the OSS bucket that you want to use and click **Select**.

Parameter	Description
Endpoint	Select the endpoint of the OSS bucket. ○ If the OSS bucket and the ECS instance are located in different regions, select **Public Endpoint**. ○ If the OSS bucket and the ECS instance are located in the same region, set the value based on the network type of the cluster. ■ If the cluster uses a virtual private cloud (VPC), select **VPC Endpoint**. ■ If the cluster uses the classic network, select **Internal Endpoint**.
Label	Attach labels to the PV.

7. Click **Create**.

Create a PVC

1. Log on to the ACK console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click the name of the cluster or click **Details** in the **Actions** column. The details page of the cluster appears.
4. In the left-side navigation pane of the details page, choose **Volumes > Persistent Volumes**.
5. On the **Persistent Volume Claims** tab, click **Create** in the upper-right corner.
6. In the **Create PVC** dialog box, configure the parameters.

Parameter	Description
PVC Type	You can select Cloud Disk, NAS, or OSS. In this example, select OSS.
Name	The name of the persistent volume claim (PVC). The name must be unique in the namespace.
Allocation Mode	In this example, select Existing Volumes. **Note** If no PV is created, you can set **Allocation Mode** to **Create Volume**.
Existing Volumes	Click **Select PV**. Find the PV that you want to use and click **Select** in the Actions column.
Capacity	The capacity of the PVC. **Note** The capacity of the PVC cannot exceed the capacity of the PV.

7. Click **Create**.After you create the PVC, you can find it on the Persistent Volume Claims tab. In this example, a PVC named csi-oss-pvc is created and a PV is bound to the PVC.

3.2.3. Mount a CPFS to an ACK cluster

This topic describes how to mount a Cloud Paralleled File System (CPFS) to an Alibaba Cloud Container Service for Kubernetes (ACK) cluster. Deep Learning Containers (DLC) allows you to store source data in CPFSs.

Context

You can use a pair of persistent volume (PV) and persistent volume claim (PVC) to mount a CPFS to an ACK cluster. However, you cannot directly create a pair of PV and PVC for a CPFS in the ACK console. You must create a YAML file to declare a PV and PVC.

Procedure

1. Run the following command to check whether the flexvolume-cpfs plug-in is installed:

```
$ kubectl get pods -n kube-system | grep flexvolume-cpfs
flexvolume-cpfs-dcldf                1/1   Running  0      98m
flexvolume-cpfs-dkrt8                1/1   Running  0      98m
```

   If the flexvolume-cpfs plug-in is not installed, install the plug-in first. For more information, see Use CPFS volumes in ACK clusters.

2. Create a PV.
   i. Create a *pai-deeplearing-cpfs-pv.yaml* file, and copy the following content to the file:

```
apiVersion: v1
kind: PersistentVolume
metadata:
name: pai-deeplearning-cpfs
labels:
 alicloud-pvname: pai-deeplearning-cpfs
spec:
capacity:
 storage: 1000Gi
accessModes:
 - ReadWriteMany
flexVolume:
 driver: "alicloud/cpfs"
 options:
  server: "cpfs-****-0th1.cn-shenzhen.cpfs.nas.aliyuncs.com@tcp:cpfs-****-mdm1.cn-shenzhen.cpfs.nas.aliyuncs.com@tcp"
  fileSystem: "****"
  subPath: "/"
  options: "rw"
```

   ii. Log on to the NAS console. On the **File System List** page, click the ID of the target CPFS in the **File System ID/Name** column.
   iii. On the **Basic Information** page, check server and fileSystem next to **Mount Target**.
   iv. Set the server and fileSystem parameters in the *pai-deeplearing-cpfs-pv.yaml* file to the displayed values.

v. Run the following command to create a PV:

```
kubectl create -f pai-deeplearing-cpfs-pv.yaml
```

3. Create a PVC.

i. Create a *pai-deeplearing-cpfs-pvc* file and then copy the following content to the file:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
name: pai-deeplearning-cpfs
spec:
accessModes:
 - ReadWriteMany
resources:
 requests:
  storage: 1000Gi
selector:
 matchLabels:
  alicloud-pvname: pai-deeplearning-cpfs
```

> **Note** The value of alicloud-pvname must be the same as that of alicloud-pvname in the *pai-deeplearing-cpfs-pv.yaml* file.

ii. Run the following command to create a PVC:

```
kubectl create -f pai-deeplearing-cpfs-pvc
```

4. Run the following commands to check whether the PV and PVC are created:

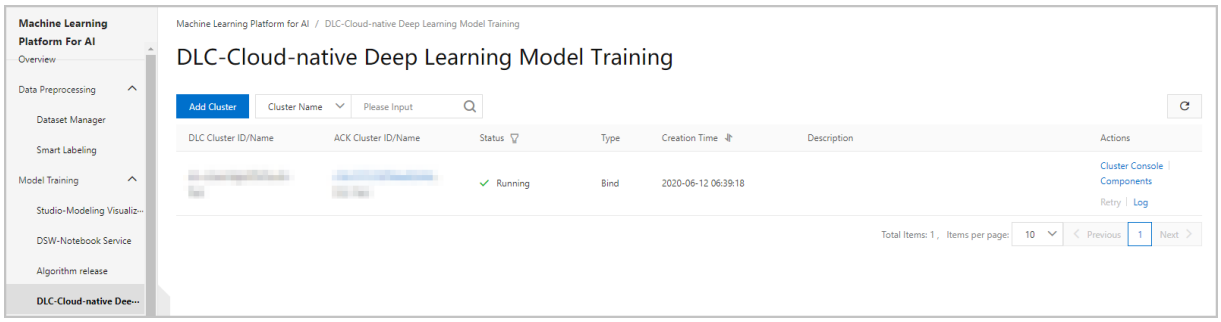
```
$ kubectl get pv | grep pai-deeplearning-cpfs
pai-deeplearning-cpfs   1000Gi   RWX       Retain      Bound   default/pai-deeplearning-cpfs
152m
$ kubectl get pvc | grep pai-deeplearning-cpfs
pai-deeplearning-cpfs  Bound   pai-deeplearning-cpfs  1000Gi   RWX
```

5. Mount the CPFS to an Elastic Compute Service (ECS) instance deployed in the same region, and upload data to the CPFS. For more information, see Mount a file system.

4.Manage clusters

4.1. Overview

This topic describes how to manage clusters on the Deep Learning Containers (DLC) platform and how to manage deep learning jobs in DLC Dashboard.



Column name	Description
DLC Cluster ID/Name	Information about a DLC cluster, including the ID and name of the DLC cluster.
ACK Cluster ID/Name	Information about a cluster of Alibaba Cloud Container Service for Kubernetes (ACK) that has been added to DLC. You can click the ID of an ACK cluster in the **ACK Cluster ID/Name** column to log on to the ACK console.
Status	• **Initializing**: DLC is initializing the ACK cluster. • **Deploying**: DLC is deploying components in the ACK cluster. • **Running**: DLC has deployed components in the ACK cluster. The ACK cluster is running. • **Deployment Failed**: DLC fails to deploy components in the ACK cluster. You can click **Log** in the **Actions** column to check the causes. You can also click **Retry** in the **Actions** column to deploy the components again.
Type	Indicates how the ACK cluster is added to DLC. • **Bind**: The ACK cluster is created in the ACK console and then added to DLC. • **Create**: The ACK cluster is created from the DLC platform.
Creation Time	The time when the DLC cluster is created.
Description	The description of the DLC cluster that you have entered after you click **Add Cluster**.

Column name	Description
Actions	• **Cluster Console**: Log on to DLC Dashboard, where you can submit training jobs and view progress of training jobs. • **Components**: View DLC components that have been deployed in the ACK cluster. • **Logs**: View log data generated when the system adds the ACK cluster and deploys the DLC components. • **Retry**: If the **Status** column displays **Deployment Failed**, you can click **Retry** to deploy the DLC components again.

4.2. Add an ACK cluster

This topic describes how to add a cluster of Alibaba Cloud Container Service for Kubernetes (ACK) to Deep Learning Containers (DLC).

Prerequisites

- The public-facing SLB ingress controller is installed in the ACK cluster. For more information, see Install the public-facing SLB ingress controller.
- The required Resource Access Management (RAM) role is created for DLC. For more information, see Authorization.

Procedure

1. Log on to the Machine Learning Platform for AI console.
2. In the left-side navigation pane, choose **Model Training > DLC-Cloud-native Deep Learning Model Training**.
3. On the DLC page, click **Add Cluster**.
4. In the **Add Cluster** dialog box, set the following parameters.

Parameter	Description
Cluster Name	The name of the DLC cluster. The cluster name must be 1 to 30 characters in length, and can contain underscores (_) and hyphens (-). It must start with a letter, digit, or Chinese character.
Description	The description of the DLC cluster, which helps distinguish between different DLC clusters.
ACK Cluster	You can add an ACK cluster in the following ways: ◦ Select an existing **ACK cluster** from the **ACK Cluster** drop-down list. ◦ Click **Create Cluster** next to **ACK Cluster** to log on to the ACK console and create an ACK cluster. For more information, see Create a dedicated Kubernetes cluster.

5. Click **OK**.
   The newly added cluster is then displayed in the cluster list of the DLC platform. After the status of the cluster in the **Status** column changes from **Deploying** to **Running**, you can proceed with

deep learning jobs.

5.Manage deep learning jobs

5.1. Images

Deep Learning Containers (DLC) allows you to use public images or custom images to run deep learning jobs.

public image

DLC provides a variety of public images that support different types of resources, Python versions, and deep learning frameworks (TensorFlow and PyTorch). The following table lists the supported public images.

Variable	Description	Valid value
region	Endpoints of supported regions	• cn-hangzhou: China (Hangzhou) • cn-shanghai: China (Shanghai) • cn-beijing: China (Beijing) • cn-shenzhen: China (Shenzhen)
type	Supported resource types	• cpu • gpu • mkl_cpu: Math Kernel Library (MKI-DNN)
python	Supported Python versions	• py2: Python 2 • py3: Python 3

The following lists show the types of resources and Python versions supported by different deep learning frameworks:

> **Note** When you use an image, you must replace ${Variable} with the corresponding value listed in the preceding table.

- **TensorFlow 1.12** (CUDA version: nvidia-cuda-10.0-cudnn7)
  - Supported regions: China (Hangzhou), China (Shanghai), China (Beijing), and China (Shenzhen)
  - Supported Python versions: Python 2 and Python 3
  - Supported resource types: CPU, GPU, and MKI_CPU
  - Image path: `registry.${region}.aliyuncs.com/pai-dlc/pai-tensorflow-training:1.12-${type}-${python}`
  - Example: `registry.cn-beijing.aliyuncs.com/pai-dlc/pai-tensorflow-training:1.12-cpu-py2`

    In this example, TensorFlow 1.12, China (Beijing), CPU, and Python 2 are specified.
- **TensorFlow 1.15** (CUDA version: nvidia-cuda-10.0-cudnn7)
  - Supported regions: China (Hangzhou), China (Shanghai), China (Beijing), and China (Shenzhen)
  - Supported Python versions: Python 2 and Python 3
  - Supported resource types: GPU

  - Image path: `registry.${region}.aliyuncs.com/pai-dlc/pai-tensorflow-training:1.15-${type}-${python}`
  - Example: `registry.cn-beijing.aliyuncs.com/pai-dlc/pai-tensorflow-training:1.15-gpu-py2`

    In this example, TensorFlow 1.15, China (Beijing), GPU, and Python 2 are specified.

- **TensorFlow 2.2** (CUDA version: nvidia-cuda-10.1-cudnn7)
  - Supported regions: China (Hangzhou), China (Shanghai), China (Beijing), and China (Shenzhen)
  - Supported Python versions: Python 3
  - Supported resource types: CPU and GPU
  - Image path: `registry.${region}.aliyuncs.com/pai-dlc/pai-tensorflow-training:2.2-${type}-${python}`
  - Example: `registry.cn-beijing.aliyuncs.com/pai-dlc/pai-tensorflow-training:2.2-cpu-py3`

    In this example, TensorFlow 2.2, China (Beijing), CPU, and Python 3 are specified.

- **PyTorch 1.3** (CUDA version: nvidia-cuda-10.0-cudnn)
  - Supported regions: China (Hangzhou), China (Shanghai), China (Beijing), and China (Shenzhen)
  - Supported Python versions: Python 2 and Python 3
  - Supported resource types: CPU, GPU, and MKI_CPU
  - Image path: `registry.${region}.aliyuncs.com/pai-dlc/pai-pytorch-training:1.3-${type}-${python}`
  - Example: `registry.cn-beijing.aliyuncs.com/pai-dlc/pai-pytorch-training:1.3-gpu-py3`

    In this example, PyTorch 1.3, China (Beijing), GPU, and Python 3 are specified.

- **PyTorch 1.5** (CUDA version: nvidia-cuda-10.1-cudnn7)
  - Supported regions: China (Hangzhou), China (Shanghai), China (Beijing), and China (Shenzhen)
  - Supported Python versions: Python 2 and Python 3
  - Supported resource types: CPU, GPU, and MKI_CPU
  - Image path: `registry.${region}.aliyuncs.com/pai-dlc/pai-pytorch-training:1.5-${type}-${python}`
  - Example: `registry.cn-beijing.aliyuncs.com/pai-dlc/pai-pytorch-training:1.5-gpu-py3`

    In this example, PyTorch 1.5, China (Beijing), GPU, and Python 3 are specified.

Custom images

DLC allows you to use custom images. For more information about how to pull a custom image, see Basic operations on Docker images. The basic Docker image provided by DLC is: *reg.docker.alibaba-inc.com/dlc/mirror:nvidia-cuda-10.0-cudnn7-devel-ubuntu16.04*.

5.2. Manage jobs in DLC Dashboard

After you add clusters of Alibaba Cloud Container Service for Kubernetes (ACK) to Deep Learning Containers (DLC), the clusters are installed with DLC Dashboard. You can manage deep learning jobs in DLC Dashboard.

Prerequisites

An ACK cluster is added to DLC. For more information, see Add an ACK cluster.

Context

DLC Dashboard allows you to manage TensorFlow jobs created based on public images. If you want to manage other types of jobs, you must use Arena. For more information, see Arena official documentation.

Submit a job

1. Log on to **DLC Dashboard**.
   i. Log on to the Machine Learning Platform for AI console.
   ii. In the left-side navigation pane, choose **Model Training > DLC-Cloud-native Deep Learning Model Training**.
   iii. On the DLC page, find the target cluster, and click **Cluster Console** in the **Actions** column.
2. In the left-side navigation pane of **DLC Dashboard**, click **Submit Job**.
3. On the **Submit Job** page, set the following parameters.

Section	Parameter		Description
Basic Information	**Job Name**		The name of the training job. The job name must be 2 to 30 characters in length and start with a lowercase letter.
	Type		By default, **TensorFlow** is selected and cannot be changed.
Job Details	**Source Code**		Set the following parameters based on the path where your source code is stored: ◦ If your source code is stored in a repository, select **Repository**, and specify **Repository Address** and **Branch**. **Note** DLC automatically downloads the source code to the directory */workspace*. Therefore, your account must be granted the permissions to access the repository. ◦ If your source code is stored in a volume mounted to the ACK cluster, select **Mounted Volume**. Then, select the volume from the **PVCs** list.
	Command		Python commands are supported. You can pass the path of the training dataset (such as *data_dir*) as a parameter to the entry function in the source code.
		Number of Workers	Specify the number of nodes to run the job: ◦ For a standalone job, you can select the default node **Worker**. ◦ For a distributed job, click **Add Node** on the right side of **Worker**, and select **PS** from the drop-down list. **Note** Parameter Server (PS) nodes do not support GPU resources.

Section	Worker Parameter		Description
		Image	Select a public image. Make sure that the **PS** and **Worker** nodes use the same image. This means that the images for both types of nodes must use the same TensorFlow and Python versions. However, the images can use different CPU and GPU resources.
		Resource	Specify **CPU vCores**, **Memory**, and **GPUs**. **Note** The value of **GPUs** must not be greater than the number of GPUs provided by the ACK cluster.

4. (Optional)On the **Submit Job** page, configure the following optional parameters as needed.

Section	Description
Environment Variables	Define environment variables in key-value pairs. You can refer to these environment variables in the source code.
Storage Configuration	If your training data is stored in the volume mounted to your ACK cluster, you can add the logic for retrieving the training data to your source code. DLC Dashboard allows you to bind the volume mounted to your ACK cluster to your training job. Then, you can pass the path where your training data is stored as a parameter to the entry function of the training job. **Note** The volume mounted to your ACK cluster must be in the `pai-dlc-user namspace` namespace. Otherwise, the volume is not displayed in the **Storage Configuration** list.

5. Click **Submit** in the lower-right corner of the page.

Query jobs

DLC Dashboard allows you to query jobs by **name**, **time range**, and **status**.

1. Log on to **DLC Dashboard**.
   i. Log on to the Machine Learning Platform for AI console.
   ii. In the left-side navigation pane, choose **Model Training > DLC-Cloud-native Deep Learning Model Training**.
   iii. On the DLC page, find the target cluster, and click **Cluster Console** in the **Actions** column.
2. In the left-side navigation pane of **DLC Dashboard**, click **Jobs**.
3. On the **Jobs** page, specify **Time Range** and click **Search**.
4. In the **Job List** on the **Jobs** page, click the name of a job in the **Name** column.
5. Then, you can check details about the job on the Job Details page.

5.3. Manage jobs in Arena

Arena is a Kubernetes-based command-line tool empowered by AI. Deep Learning Containers (DLC) allows you to use clusters of Alibaba Cloud Container Service for Kubernetes (ACK) to train models in Arena.

Step 1: Install the client

1. Log on to the Arena website, and download the Arena installation package. For macOS, download arena-installer-xxx-xxx-darwin-amd64.tar.gz. For Linux, download arena-installer-xxx-xxx-linux-amd64.tar.gz.
2. Run the following command to install the client:

   ```
   tar -xvf arena-installer-xxx-xxx.tar.gz
   cd arena-installer
   sudo ./install.sh
   ```

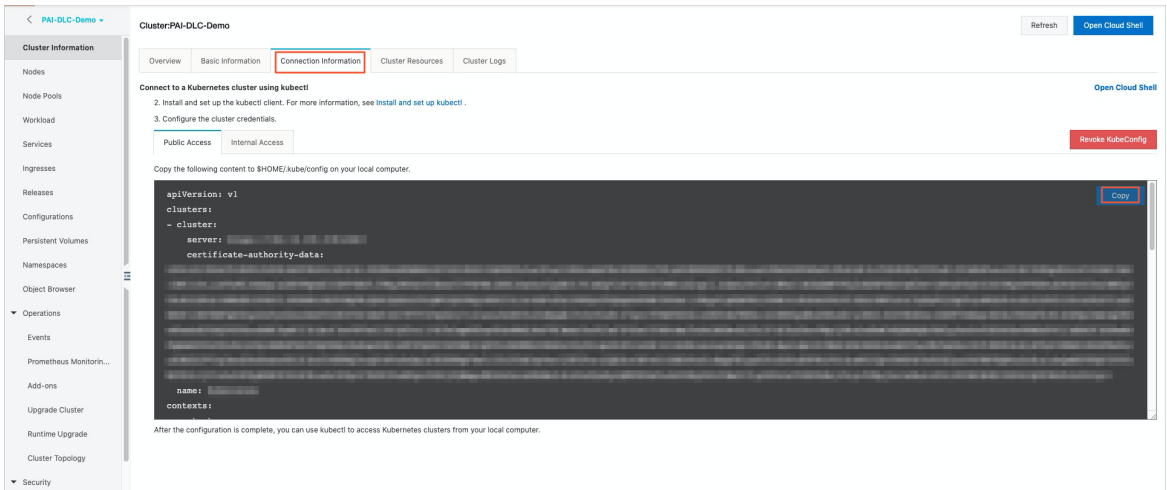
   Replace arena-installer-xxx-xxx.tar.gz with the actual name of the package.
3. Run the following command to check whether the client is installed:

   ```
   arena version
   ```

Step 2: Configure KubeConfig

If you want to remotely connect to an ACK cluster and submit a job, you must include the KubeConfig of the ACK cluster in *$HOME/.kube/config*.

1. Log on to the Machine Learning Platform for AI console.
2. In the left-side navigation pane, choose **Model Training > DLC-Cloud-native Deep Learning Model Training**.
3. On the DLC page, click the ID of your ACK cluster in the **ACK Cluster ID/Name** column.
4. On the **Connection Information** tab of the Cluster Information page, click **Copy**.



5. Create a local *.kube/config* file, and paste the configurations to the file.

   ```
   vim $HOME/.kube/config
   ```

Step 3: Submit a TensorFlow job

1. Run the command to submit a TensorFlow job.You can run one of the following commands to submit a TensorFlow job:

```
# Submit a TensorFlow job.
arena submit tfjob + paraname
```

```
# Submit a TensorFlow job.
arena submit tf + paraname
```

Replace paraname with the desired parameter. You can run the `arena submit tfjob --help` command to query all supported parameters. Required parameters are:

- --name: the name of the job.
- --image: the Docker image that is used to launch pods for training deep learning models. The image must be supported by DLC. You can select a public image or a custom image based on the region, deep learning framework, Python version, and resource type of your DLC cluster. For more information, see Images.
- --data: the directory where the source data is stored. The path must be in the format of *PVC name:directory*.

2. You can use one of the following methods to view the log of a job:
   - Run the following command:

     ```
     arena logs yourTaskName
     ```

     Replace yourTaskName with the actual name of the job.
   - For more information about how to view logs of training jobs in DLC Dashboard, see Manage jobs in DLC Dashboard.

Examples

- Standalone job

  ```
  arena submit tf \
  --name=pai-deeplearning-test-oss \
  --image=registry.cn-shanghai.aliyuncs.com/pai-dlc/pai-tensorflow-training:1.12-cpu-py2 \
  --data=pai-deeplearning-oss:/training_dir/ \
  "python /training_dir/code/main.py --max_steps=10000 --data_dir=/training_dir/data/"
  ```

  - --name: the name of the job. After you submit the job, you can run the `arena logs ${name}` command to view the log of the job.
  - --image: the path of the image. Example: `registry.cn-shanghai.aliyuncs.com/pai-dlc/pai-tensorflow-training:1.12-cpu-py2`.
  - --data: the directory where the source data is stored. Example: *pai-deeplearning-oss:/training_dir*. pai-deeplearning-oss represents the persistent volume claim (PVC) created for the ACK cluster. */training_dir* represents a directory on a pod. Make sure that the specified PVC is mounted to */training_dir*.
  - python /training_dir/code/main.py --max_steps=10000 --data_dir=/training_dir/data/: the command to be executed by the pod. */training_dir/code/* represents the directory where the source code is stored in the Object Storage Service (OSS) bucket. --max_steps and --data_dir correspond to the FALGS.max_steps and FLAGS.data_dir parameters in main.py.

- Distributed job

```
arena submit tf \
--name=pai-deeplearning-dist-test-nas \
--workers=2 \
--worker-image=registry.cn-shanghai.aliyuncs.com/pai-dlc/pai-tensorflow-training:1.12-cpu-py2 \
--ps=1 \
--ps-image=registry.cn-shanghai.aliyuncs.com/pai-dlc/pai-tensorflow-training:1.12-cpu-py2 \
--data=pai-deeplearning-nas:/training_dir/ \
"python /training_dir/code/dist-main.py --max_steps=10000 --data_dir=/training_dir/data/"
```

- --name: the name of the job. After you submit the job, you can run the `arena logs ${name}` command to view the log of the job.
- --workers: the number of worker nodes.
- --worker-image: the path of the image for worker nodes.
- --ps: the number of PS nodes.
- --ps-image: the path of the image for PS nodes.
- --data: the directory where the source data is stored. Example: *pai-deeplearning-nas:/training_dir/*. pai-deeplearning-nas represents the PVC created for the ACK cluster. */training_dir* represents a directory on a pod. Make sure that the specified PVC is mounted to */training_dir*.
- python /training_dir/code/dist-main.py --max_steps=10000 --data_dir=/training_dir/data/: the command to be executed by the pod. */training_dir/code/dist-main.py* represents the directory where the source code is stored in the NAS file system. --max_steps and --data_dir correspond to the FALGS.max_steps and FLAGS.data_dir parameters in main.py.

- Distributed job (GPU-accelerated)

```
arena submit tf \
--name=pai-deeplearning-gpu-dist-test-oss \
--gpus=1 \
--workers=2 \
--worker-image=registry.cn-shanghai.aliyuncs.com/pai-dlc/pai-tensorflow-training:1.12-gpu-py2 \
--ps=1 \
--ps-image=registry.cn-shanghai.aliyuncs.com/pai-dlc/pai-tensorflow-training:1.12-cpu-py2 \
--data=pai-deeplearning-nas:/training_dir/ \
"python /training_dir/code/dist-main.py --max_steps=10000 --data_dir=/training_dir/data/"
```

- --name: the name of the job. After you submit the job, you can run the `arena logs ${name}` command to view the log of the job.
- --gpus: the number of GPUs allocated to each worker node. The value of this parameter must not be greater than the number of GPU-accelerated nodes in your ACK cluster.
- --workers: the number of worker nodes.
- --worker-image: the path of the GPU image for worker nodes.
- --ps: the number of PS nodes.
- --ps-image: the path of the CPU image for PS nodes.
- --data: the directory where the source data is stored. Example: *pai-deeplearning-oss:/training_dir/*. pai-deeplearning-oss represents the PVC created for the ACK cluster. */training_dir* represents a directory on a pod. Make sure that the specified PVC is mounted to */training_dir*.

- python /training_dir/code/dist-main.py --max_steps=10000 --data_dir=/training_dir/data/: the command to be executed by the pod. */training_dir/code/dist-main.py* represents the directory where the source code is stored in the OSS bucket. --max_steps and --data_dir correspond to the FALGS.max_steps and FLAGS.data_dir parameters in main.py.