Lenovo

# LiCO 6.4.0
# User Guide

**Twelfth Edition (June 2022)**

# Contents

# Chapter 1. Overview

## Introduction to LiCO

Lenovo Intelligent Computing Orchestration (LiCO) is an infrastructure management software for high-performance computing (HPC) and artificial intelligence (AI). It provides features like cluster management and monitoring, job scheduling and management, cluster user management, account management, and file system management.

With LiCO, users can centralize resource allocation in one supercomputing cluster and carry out HPC and AI jobs simultaneously. Users can perform operations by logging in to the management system interface with a browser, or by using command lines after logging in to a cluster login node with another Linux shell.

## Features of LiCO

- **Cluster resource monitoring**: LiCO provides a dashboard to monitor the usage of cluster resources, including CPU, memory, storage, and network.
- **Job template storage**: LiCO provides multiple job templates, including HPC and AI job templates, which help users submit jobs from Web pages with convenience.
- **Customized templates**: Users can create their own job templates to support other HPC and AI applications.
- **Job management and monitoring**: Users can directly view and manage the status and results of jobs. Various common schedulers and a wide range of job types are supported (including AI jobs such as TensorFlow and Caffe).
- **E2E training**: Users can train image classification models without coding. LiCO also provides E2E support for training, such as dataset management, topology management, and pre-trained model management.
- **User management and billing**: LiCO manages both local and domain users through the same interface. It supports user top-ups and chargebacks, and offers the ability to set billing groups and fees.
- **Customizations**: A range of customizations are available, such as enterprise job template customization, report customization, and 3D server visualization.
- **Container image management**: LiCO provides system container images for every supported AI framework. Users can upload private container images and run AI or HPC jobs on them.

## Terminology

- **Computer cluster**: a general reference to a collection of server resources including management nodes, login nodes, and computing nodes
- **Job**: a series of commands in sequence intended to accomplish a particular task
- **Job status**: the status of a job in the scheduling system, such as waiting, in queue, on hold, running, suspended, or completed
- **Node status**: the status of a node, such as idle, busy, or off
- **Job scheduling system**: the distributed program in control of receiving, distributing, executing and registering jobs, also referred to as the operation scheduler or simply scheduler
- **Management node**: the server in a cluster running management programs such as job scheduling, cluster management and user billing
- **Login node**: the server in a cluster to which users can log in via Linux and conduct operations
- **Computing node**: the server in a cluster for executing jobs

- **User group**: a set of users for which the system has defined an access control policy, so that all users in the same user group have access to the same set of cluster resources
- **Billing group**: a group of cluster users that are to be billed under one account, also referred to as a billing account. A billing account can be made up of a single user or multiple users.
- **NGC Image**: NVIDIA Container Runtime for Docker, also known as nvidia-docker. It supports GPU-based applications that are portable across multiple machines. This is achieved through the use of Docker containers.

## Prerequisite

- LiCO currently supports Slurm as the scheduler. The commands for Slurm in this Guide are not applicable to other schedulers.
- The paths for jobs involved in this Guide do not support spaces or other special characters.

## Operating environment

**Cluster server:**

Lenovo ThinkSystem servers

**Operating system**

- Red Hat Enterprise Linux (RHEL) 8.5/Rocky Linux 8.5
- SUSE Linux Enterprise Server (SLES) 15 SP3

**Client requirements:**

- Hardware: CPU of 2.0 GHz or above, memory of 8 GB or above
- Browser: Chrome (V 62.0 or higher) or Firefox (V 56.0 or higher) recommended
- Display resolution: 1280 x 800 or above

# Chapter 2. Basic operations

**Note:** Instructions in this chapter are primarily based on the management system interface. Command line users can refer to "" on page and "References for Slurm commands" on page 94 for further instructions.

## Log in

The client must have direct access to the cluster login node.

Step 1. Open a browser.

Step 2. Type the IP address for the cluster's login node, such as `https://10.220.112.21`.

Step 3. Type the username and password.

Step 4. Click **Log in**.

## Log out

Step 1. Place your cursor over  in the upper-right corner of the page.

Step 2. Click  .

Step 3. Click **Confirm**.

You have logged out from LiCO.

## Get current version information

Step 1. Place your cursor over  in the upper-right corner of the home page.

Step 2. Click .
A page that shows the current version information is displayed.

Step 3. Click a menu item to get required information.

Click **User Agreement** to get the user agreement.

Click **Third party licenses** to obtain the third party licenses.

## Change the password

Step 1. Place your cursor over  in the upper-right corner.

Step 2. Click  .
A dialog is displayed for you to change the password.

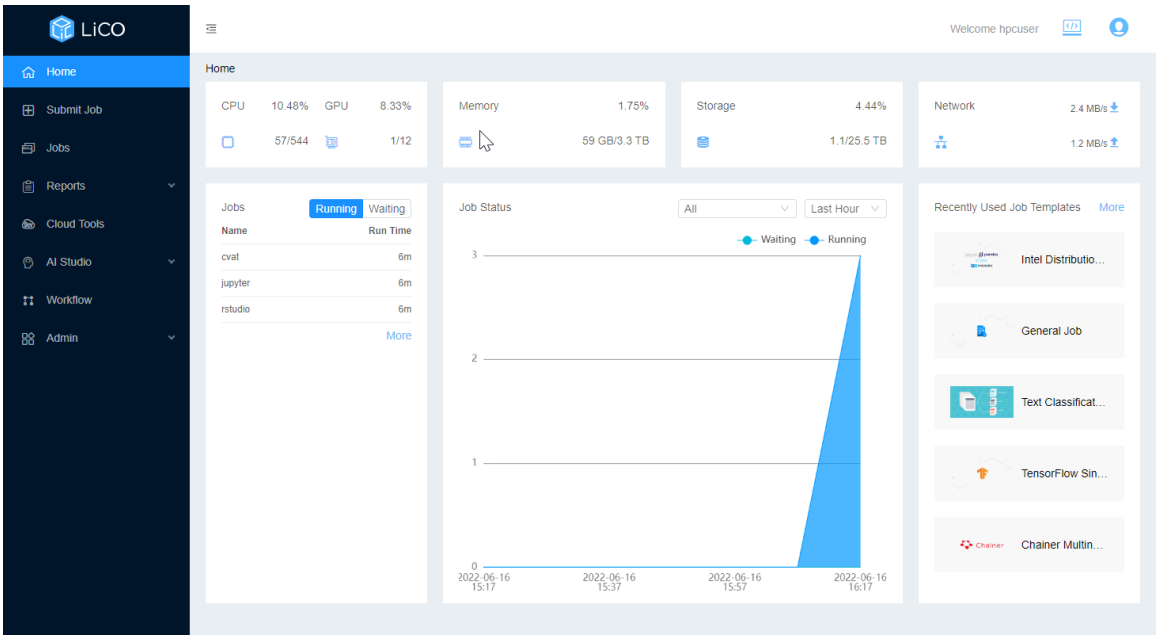
Step 3. Type the current password, and then type the new password twice.

Step 4. Click **OK**.

## View cluster resources and job status

Select **Home** from the left navigation pane.

The cluster overview page is displayed.




## Elements on the cluster overview page

- **CPU**: Shows the CPU usage in the cluster, with the number of occupied CPU cores and the total number of CPU cores in the cluster.
- **Memory**: Shows the memory usage in the cluster, with the used memory and the total memory in the cluster.
- **Storage**: Shows the storage usage in the cluster, with the used storage and total storage.
- **Network**: Shows the upload and download rates.
- **Jobs**: Shows the information about jobs that the current user has submitted to the **Running** or **Waiting** queue. Switch between **Running** and **Waiting** to view the names of current jobs and their running or waiting time. Click **More** to go to the task details page to view more detailed information about the execution of the jobs.
- **Job Status**: Shows the status of all jobs submitted by the current user. Jobs can be viewed by queue or time period. When sorted by queue, the names of all queues in a cluster are listed. Time period options include **Last hour**, **Last 1 day**, **Last 7 days**, and **Last 30 days**. In the figure, the user can choose historical jobs that are running or waiting, and view the number of jobs in the running status at a specific point in time.
- **Recently Used Job Templates**: Shows the job templates that were recently used by the user and can be leveraged.

# System tools

## Manage system tools page

Step 1. Click the system tools icon  in the upper-right corner of the home page. The File Manage page and the Console page will be displayed.

Step 2. Do one of the following:

- To display the File Manage area or the Console area, click the maximization icon  in the upper-right corner of the target area.
- To hide the File Manage area or the Console area, click the hidden icon  or double-click the scrollbar of the File Manage area or the Console area.
- To expand the hidden File Manage area or Console area, click the expansion icon  on the right of the File Manage area or the expansion icon  on the left of the Console area.
- To change the size of the File Manage area and the Console area, drag the scroll bar vertically.

  **Note:** The area-to-interface ratio is 30% to 70%.

## Manage files

### Create a folder

Step 1. Click the system tools icon  in the upper-right corner of the home page.

Step 2. Right-click in the blank area of the File Manage page, and select **New folder** from the shortcut menu.
A new folder is created, with “NewFolder” as its default name.

### Rename a folder

Step 1. Click the system tools icon  in the upper-right corner of the home page.

Step 2. Right-click the target folder, and then select **Rename**.

Step 3. Type the new folder name in the text box.

### Preview an image

Step 1. Click the system tools icon  in the upper-right corner of the home page.

Step 2. Right-click the target image, and then select **Preview** from the shortcut menu.

### Archive files

Step 1. Click the system tools icon  in the upper-right corner of the home page.

Step 2. Right-click a file or folder, place the cursor over **Create archive** from the shortcut menu, and then select the compression format.

## Extract an archived file

Step 1. Click the system tools icon  in the upper-right corner of the home page.

Step 2. Right-click a target archived file (compressed in TAR, ZIP, or GZIP format), and then select **Extract files from archive** from the shortcut menu.

## Upload files

Step 1. Click the system tools icon  in the upper-right corner of the home page.

Step 2. Double-click a folder to open it.

Step 3. Right-click in the blank area, and select **Upload files** from the shortcut menu.

Step 4. Select one or more local files to upload using either of the following methods:

- Drag and drop a file or files into the dotted dialog.
- Click **Select files to upload** at the bottom of the Upload files window.

Successfully uploaded files are available on the File Manage page.

## Copy and paste files

Step 1. Click the system tools icon  in the upper-right corner of the home page.

Step 2. Right-click a file or folder, and then select **Copy**.

Step 3. Right-click in the blank area, and then select **Paste** from the shortcut menu.
The copied file or folder has been pasted in the same folder.

**Notes:**

- Shortcut keys Ctrl+C and Ctrl+V are also applicable as copy and paste operations.
- You can copy a file or folder from one folder to a different folder.
- If the destination folder has a file or folder with the same name as the one being copied, a pop-up dialog will be displayed. You may then determine whether to overwrite the existing file or folder with the new one.

## Move files

Step 1. Click the system tools icon  in the upper-right corner of the home page.

Step 2. Right-click a file or folder, and then select **Cut**.

Step 3. Select a destination folder, right-click in the blank area, and then select **Paste** from the shortcut menu.
The selected file or folder has been moved to the destination folder.

**Note:** If the destination folder has a file or folder with the same name as the one being copied, a pop-up dialog will be displayed. You may then determine whether to overwrite the existing file or folder with the new one.

## Duplicate files

Step 1. Click the system tools icon  in the upper-right corner of the home page.

Step 2. Right-click a file or folder, and then select **Duplicate**.

### Download files

Step 1. Click the system tools icon  in the upper-right corner of the home page.

Step 2. Right-click a file, and then select **Download**.

### Edit files

Step 1. Click the system tools icon  in the upper-right corner of the home page.

Step 2. Right-click a file, and then select **Edit**.

Step 3. After editing the file, do any of the following:

- **Save**: Save the changes.
- **Save & Close**: Save the changes and exit from the edit page.
- **Save As**: Save the current file as another, without changing the original file.
- **Cancel**: Exit from the edit page. You can click **Save** to save the changes before clicking **Cancel** to exit, or directly exit without saving the changes.

### Edit files in batch

**Note:** This feature is not supported in the Select Folder page.

Step 1. Click the system tools icon  in the upper-right corner of the home page.

Step 2. Right-click the target file and select **Edit**. Repeat this step until all target files are open.

**Note:** Up to ten files can be edited at the same time.

Step 3. Do one of the following:

- To view the file list, hover the mouse over  in the upper-right corner of the home page.
- To close the target file, click  of the target file.
- To display the hidden file, click the file name of the target file in the file list.

## Manage console

Step 1. Click the system tools icon  in the upper-right corner of the home page.

Step 2. On the Console page, click **Connect** or  on the upper-left corner. The Connect page is displayed.

Step 3. On the Connect page, input or select host name from the drop-down list, and select terminal type from the drop-down list.

Step 4. Click **Connect**. The Console window is displayed.

Step 5. In the Console window, input user name and password to log in to the target console.

**Note:** Up to ten consoles can be connected at the same time. Users can click the tab to locate to the target console, or click  to close the target console.

# Container images

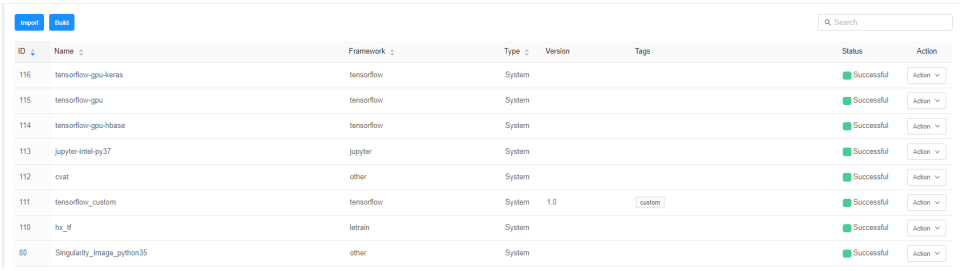
LiCO runs all AI jobs within a container. The system supports the Singularity container platform, with different AI job templates running on different container images.

In addition to providing a certain number of basic container images, LiCO also allows users to upload customized container images. LiCO 5.2.0 and later versions support running jobs on NGC images. To transform an NGC image to a Singularity container image, refer to “Transform an NGC image” on page 94.

# View container images

Select **Admin → Container Images** from the left navigation pane.

The Container Images page is displayed.




The parameters on the Container Images page are described as follows:

- **Name**: self-defined container image name
- **Framework**: frame to which the container image belongs
- **Type**: type of the container image, which can be **private** or **system**. The value **private** means that the container image is created by yourself, and the value **system** means that the container image is created by the system administrator.
- **Version**: self-defined container image version
- **Tag**: self-defined container image tag
- **Action**: action on the container image. For system container images, available actions are **Browse** and **Download**; for private container images, available actions are **Edit**, **Browse**, **Delete**, **Download**, and **Reupload**.

# Build a container image

On the container image management page, click **Build** above the container image list.
A page for building an image is displayed.

Users can build a container image from any of the following sources:

- “Build a container image from a Docker registry” on page 8
- “Build a container image from a Singularity library” on page 9
- “Build a container image from a Singularity definition file” on page 9
- “Build a container image from a system or private image” on page 10

## Build a container image from a Docker registry

Step 1. Select **Admin → Container Images** from the left navigation pane.

Step 2. Click **Build** above the container image list.

Step 3. Fill in the required information.

- **Name**: self-defined container image name.
- **Workspace**: working directory of the container image.
- **Source**: Select **Docker Registry**, indicating that the container image is built from Docker Hub.
- **Image Path**: path of the container image in Docker Hub.
- **Authentication**: Enter the username and password if the container image repository is a private repository.
- **Advanced**: This allows you to install Python libraries.
- **Use HTTPS**: This allows you to enable or disable HTTPS.

**Note:** To refill in the parameters, click **Reset**.

Step 4. Click **Start Build**.

**Note:** To cancel the container image that is being built, click **Cancel Build**.

After the container image is built, you can import it by clicking **Import** in the lower right corner of the build log box. Refer to "Import a container image" on page 10.

## Build a container image from a Singularity library

Step 1. Select **Admin → Container Images** from the left navigation pane.

Step 2. Click **Build** above the container image list.

Step 3. Fill in the required information.

- **Name**: self-defined container image name.
- **Workspace**: working directory of the container image.
- **Source**: Select **Singularity Library**, indicating that the container image is built from the Container Library.
- **Image Path**: path of the container image in the Container Library.
- **Advanced**: This allows you to install Python libraries.

**Note:** To refill in the parameters, click **Reset**.

Step 4. Click **Start Build**.

**Note:** To cancel the container image that is being built, click **Cancel Build**.

After the container image is built, you can import it by clicking **Import** in the lower right corner of the build log box. Refer to "Import a container image" on page 10.

## Build a container image from a Singularity definition file

Step 1. Select **Admin → Container Images** from the left navigation pane.

Step 2. Click **Build** above the container image list.

Step 3. Fill in the required information.

- **Name**: self-defined container image name.
- **Workspace**: working directory of the container image.

- **Source**: Select **Singularity Definition File**, indicating that the container image is built from a custom .def file.
- **Definition File**: path of the custom .def file.
- **Authentication**: This allows you to enable authentication, depending on the container image repository type in the .def file.
- **Use HTTPS**: This allows you to enable or disable HTTPS.

**Note:** To refill in the parameters, click **Reset**.

Step 4. Click **Start Build**.

**Note:** To cancel the container image that is being built, click **Cancel Build**.

After the container image is built, you can import it by clicking **Import** in the lower right corner of the build log box. Refer to "Import a container image" on page 10.

### Build a container image from a system or private image

Step 1. Select **Admin → Container Images** from the left navigation pane.

Step 2. Click **Build** above the container image list.

Step 3. Fill in the required information.

- **Name**: self-defined container image name.
- **Workspace**: working directory of the container image.
- **Source**: Select **System Image** or **Private Image**, indicating that the container image is built from a system or private image.
- **Image Path**: path of the container image.
- **Advanced**: This allows you to install Python libraries.

**Note:** To refill in the parameters, click **Reset**.

Step 4. Click **Start Build**.

**Note:** To cancel the container image that is being built, click **Cancel Build**.

After the container image is built, you can import it by clicking **Import** in the lower right corner of the build log box. Refer to "Import a container image" on page 10.

## Import a container image

Step 1. Select **Admin → Container Images** from the left navigation pane.

Step 2. Click **Import** above the container image list.

Step 3. Fill in the required information.

- **Name**: self-defined container image name.
- **Framework**: frame to which the container image belongs
- **Source File**: container image file selected, which must be a singularity container image file. Otherwise, the container image will fail to be created.
- **Save As**: name of the container image file. Ensure that there is no container image file with the same name in the storage path.
- **Version**: self-defined container image version.
- **Tags**: self-defined container image tag.

- **Description**: any description about the imported container image.

Step 4. Click **OK**.

## View a container image

Step 1. Select **Admin → Container Images** from the left navigation pane.

Step 2. Find the container image for which you want to view its information, and then select **Action → Browse**.

The information is described as follows:

- **Image Path**: location of the container image
- **Description**: description of the container image. If no description has been filled in, this information is not displayed.

## Download a container image

Step 1. Select **Admin → Container Images** from the left navigation pane.

Step 2. Find the container image you want to download, and then select **Action → Download**.

Step 3. Click **Browse**.

Step 4. Select a folder to save the container image.
Ensure that there is no container image file with the same name in the storage path.

Step 5. Click **OK**.

## Edit a container image

Step 1. Select **Admin → Container Images** from the left navigation pane.

Step 2. Find the private container image you want to edit, and then select **Action → Edit**.

Step 3. Edit the container image information as required.

Step 4. Click **OK**.

## Delete a container image

Step 1. Select **Admin → Container Images** from the left navigation pane.

Step 2. Find the private container image you want to delete, and then select **Action → Delete**.

Step 3. Click **OK**.

## Reupload a container image

Step 1. Select **Admin → Container Images** from the left navigation pane.

Step 2. Find the private container image you want to reupload, and then select **Action → Reupload**.

Step 3. Click **Browse** and then select a new container image file you want to upload.

Step 4. Click **OK**.

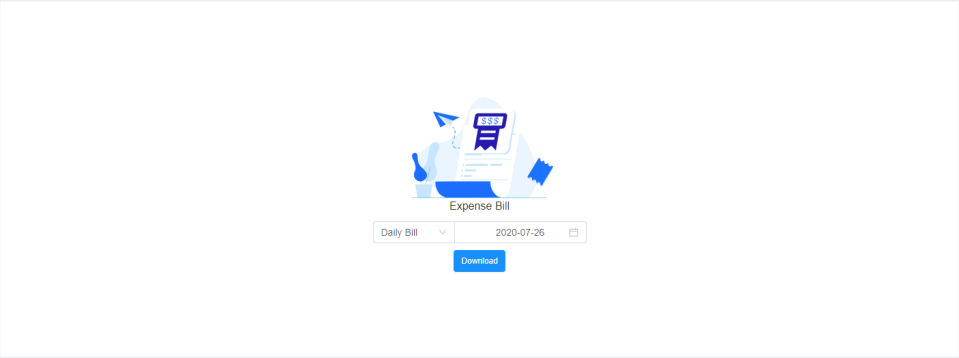
---

# Bills

LiCO 5.5.0 and later versions can bill users for jobs and storage instances. Users can download their daily and monthly bills generated automatically on the system.

Select **Admin → Bills** from the left navigation pane.

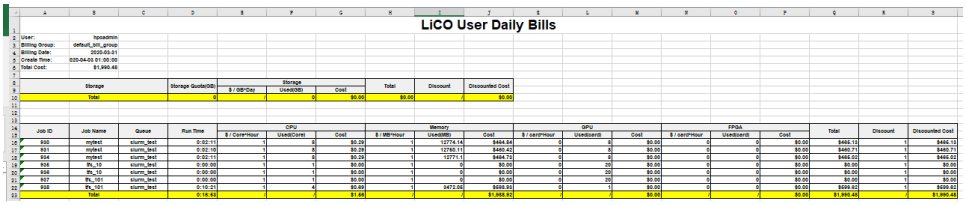
The page for download bills is displayed.




The following types of bills are available:

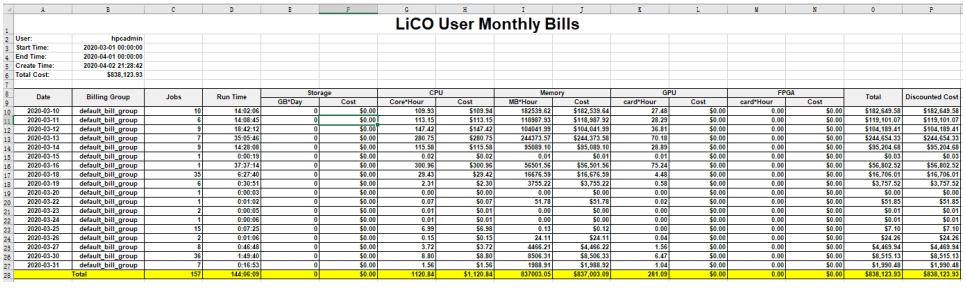
- **Daily Bill**

  **Daily Bill** enables users to check the billing information for every job and storage instance on a specific day.




- **Monthly Bill**

  **Monthly Bill** enables users to check the billing information for every day in a specific month.

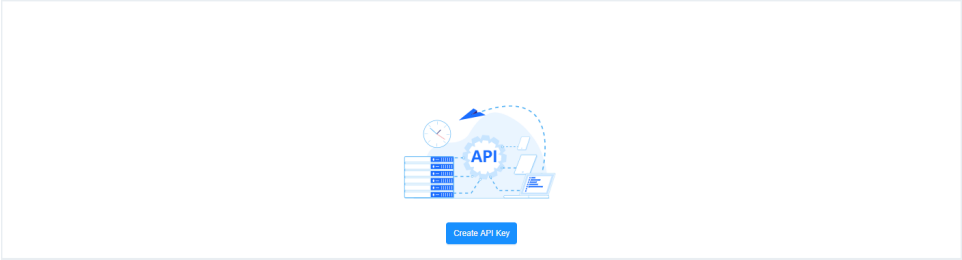



# API key

LiCO 5.3.0 and later versions provide some open application programming interfaces (APIs). To learn how to use these APIs, refer to the *LiCO 6.2.0 OpenAPI Guide*. To use these APIs, obtain an API key first.

# Create a permanent API key

Step 1. Select **Admin → API Key** from the left navigation pane.
The API Key page is displayed.




Step 2. Click **Create API Key**.

Step 3. Click **Save**.

## Create a temporary API key

Step 1. Select **Admin → API Key** from the left navigation pane.

Step 2. Click **Create API Key**.

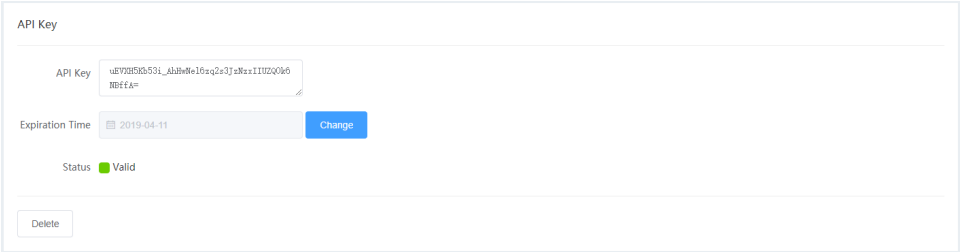
Step 3. Clear the **Unlimited** check box, and click the box below with the prompt of **Please choose a date**.

Step 4. Select a date and click **Save**.

## View an API key

Select **Admin → API Key** from the left navigation pane.

The API Key page is displayed.




## Delete an API key

Step 1. Select **Admin → API Key** from the left navigation pane.

Step 2. Click **Delete**.

Step 3. Click **OK**.

## Change a permanent API key

To change a permanent API key to a temporary one, complete the following steps:

Step 1. Select **Admin → API Key** from the left navigation pane.

Step 2. Click **Change**.

Step 3. Click the box with a calendar icon, and then select a date on the displayed date selector.

Step 4. Click **OK**.

## Change a temporary key

You can change a temporary API key to a permanent one or change its expiration time.

### Change a temporary API key to a permanent one

Step 1. Select **Admin → API Key** from the left navigation pane.

Step 2. Click **Change**.

Step 3. Click **OK**.

### Change the expiration time of a temporary API key

Step 1. Select **Admin → API Key** from the left navigation pane.

Step 2. Click **Change**.

Step 3. Clear the **Unlimited** check box, click the box with a calendar icon, and then select a date on the displayed date selector.

Step 4. Click **OK**.

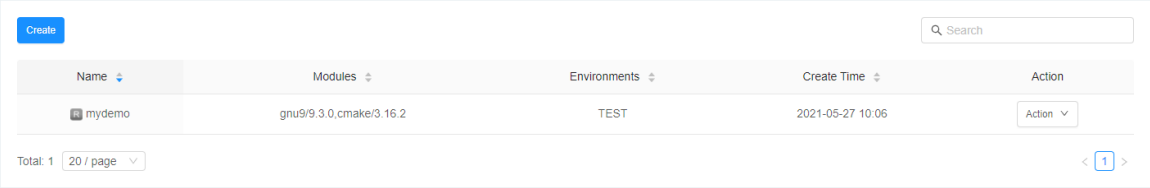
## Runtime

LiCO 5.3.0 and later versions support the runtime feature. With this feature, an isolated runtime environment can be provided for each job. Users can customize their modules and environment variables before using the runtime feature. In addition, this feature is reusable and maintainable for users’ convenience.

## View runtimes

Select **Admin → Runtime ENV** from the left navigation pane.

The Runtime page is displayed.




## Create a runtime

Step 1. Select **Admin → Runtime ENV** from the left navigation pane.

Step 2. Click **Create**.

Step 3. Fill in the required information.

- **Name**: self-defined runtime name, which is mandatory.
- **Modules**: OpenHPC computing modules
- **Environments**: environments that need to be loaded to run a job
- **Script Files**: the script files for runtime

Step 4. Click **OK**.

## Edit a runtime

Step 1. Select **Admin → Runtime ENV** from the left navigation pane.

Step 2. Find the runtime you want to edit, and then select **Action → Edit**.

Step 3. Edit the runtime name if needed.

Step 4. In the **Modules** area, click **Add**.
A page for selecting modules is displayed.

Step 5. Select the required modules and click **Confirm**.

**Note:** If Intel oneAPI is installed, the **Intel oneAPI** tab will be available, which displays all installed Intel oneAPI modules.

Step 6. Adjust the module loading order as required. To change the order of a module or delete a module, click  .

The operations are described as follows:

- **Move Up**: Move up the selected module by one place.
- **Move Down**: Move down the selected module by one place.
- **Delete**: Delete the selected module.

**Note:** To check whether the selected modules are reasonable, click **Verify** in the Edit Runtime dialog.

Step 7. In the **Environments** area, click **Add**.
The Create Environment dialog is displayed. Fill in the required information and click **Confirm**.

The parameters are described as follows:

- **Variable**: variable name, which is mandatory.
- **Value**: variable value.

Step 8. In the **Script Files** area, click **Add → Browse**. The Select File page is displayed.

Step 9. On the Select File page, select the target file and click **Select File** on the top bar.

Step 10. Adjust the script files loading order as required. To change the order of a script file or delete a script file, click  .

The operations are described as follows:

- **Move Up**: Move up the selected script file by one place.
- **Move Down**: Move down the selected script file by one place.
- **Delete**: Delete the selected script file.

Step 11. Click **OK**.

## Duplicate a runtime

Step 1. Select **Admin → Runtime ENV** from the left navigation pane.

Step 2. Find the runtime you want to duplicate, and then select **Action → Duplicate**.

Step 3. Modify the runtime name and click **OK**.

## Verify a runtime

Step 1. Select **Admin → Runtime ENV** from the left navigation pane.

Step 2. Find the runtime you want to verify, and then select **Action → Verify**.

## Delete a runtime

Step 1. Select **Admin → Runtime ENV** from the left navigation pane.

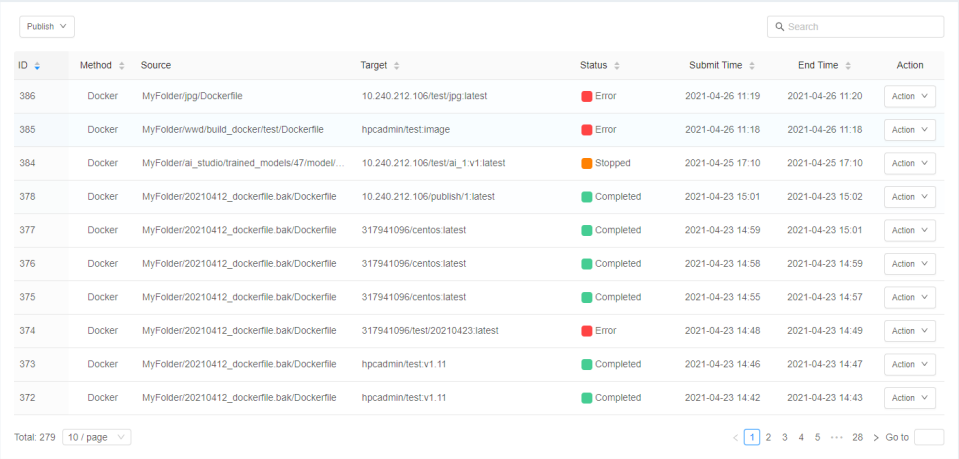
Step 2. Find the runtime you want to delete, and then select **Action → Delete**.

Step 3. Click **OK**.

# Publishing

Select **Admin → Publishing** from the left navigation pane.

The Publishing page is displayed.




This page lists all the publishing tasks and shows the publishing history of the current user.

- Publishing tasks can be sorted by **ID**, **Method**, **Target**, **Status**, **Submit Time**, or **End Time**.
- You can search for specific tasks by **ID**, **Method**, or **Target** in the search box.
- The status of a publishing task can be any of the following:
  - **Running**: The publishing task is submitted successfully and being executed.
  - **Error**: The publishing task failed to be submitted with an error in execution.
  - **Stopped**: The publishing task is actively stopped by the user.
  - **Completed**: The publishing task is normally executed and completed.

## Create a Git publishing task

Step 1. Select **Admin → Publishing** from the left navigation pane.

Step 2. Select **Publish → Git**.
A page for creating a Git publishing task is displayed.

Step 3. Fill in the required information.

- **Local Path**: Click **Browse** and select the folder that stores the files to be published.

  **Note:** Ensure that the local path does not contain any large-size file. Otherwise, the publishing task may fail to be created.
- **Repository**: Enter the address of a remote Git repository in SSH or HTTP/HTTPS format.
- **Authentication**: Authenticate the transport protocol.
  - SSH authentication is used if the repository name starts with `git@`.

  - HTTP authentication is used if the repository name starts with `http` or `https`.
- **Branch Name**: Enter a remote branch name. This parameter is optional and its default value is **master**.
- **Target Folder**: Specify a remote target folder to save the published files. This parameter is optional and can be left blank.

Step 4. Click **Submit**.
To view the logs and information of the task, select **Action → View**.

## Create a Docker publishing task

Step 1. Select **Admin → Publishing** from the left navigation pane.

Step 2. Select **Publish → Docker**.
A page for creating a Docker publishing task is displayed.

Step 3. Fill in the required information.

- **Repository**: Enter the address of a repository to publish a Docker image.
- **Tag**: Specify a tag for the Docker image in the repository.
- **Workspace**: Specify the directory that contains the content required to produce the Docker image.

  **Note:** Ensure that the local path does not contain any large-size file. Otherwise, the publishing task may fail to be created.
- **Dockerfile**: Specify the file used to produce the Docker image.
- **Pull/Push Settings**: Configure authentication for Pull (from the source repository for the base image) and Push (to the destination repository). You can specify the username and password of the remote repository, and determine whether to use HTTPS.

Step 4. Click **Submit**.
To view the logs and information of the task, select **Action → View**.

**Note:** If unexpected issues are found when you create a Docker publishing task, refer to "Publishing issues troubleshooting" on page 96.

## Republish a task

Step 1. Select **Admin → Publishing** from the left navigation pane.

Step 2. Find the task you want to republish, and then select **Action → Republish**.

Step 3. Click **OK**.

## Delete a publishing task

Step 1. Select **Admin → Publishing** from the left navigation pane.

Step 2. Find the task you want to delete, and then select **Action → Delete**.

Step 3. Click **OK**.

## Stop a publishing task

Step 1. Select **Admin → Publishing** from the left navigation pane.

Step 2. Find the publishing task you want to stop, and then select **Action → Stop**.

Step 3. Click **OK**.

# Chapter 3. Lenovo-accelerated AI

Lenovo-accelerated AI is based on the LeTrain project. LeTrain is a distributed training engine based on TensorFlow and optimized by Lenovo. Its goal is to make distributed training as easy as single GPU training and achieve linear scaling performance. Many popular models have been built into LeTrain. You can use them directly without coding. LiCO 5.3.0 and later versions support to submit Lenovo-accelerated AI jobs through APIs. To learn how to use the APIs, refer to the *LiCO 6.2.0 OpenAPI Guide*.

Select **Submit Job** from the left navigation pane, and then select the **Lenovo Accelerated AI** tab. The Lenovo Accelerated AI tab page is displayed. All the job submission tasks are performed on this page.

## Job submission – Train

## Submit an Image Classification – Train job

Step 1. In the Image Classification area, click **Train → Train**.

Step 2. Fill in the required information.

- **Job Name** (required): job name. For LiCO 6.1.0 and later versions, the initial job name will be automatically created in the format of "template name + time".
- **Workspace** (required): working directory of the job. Job output files will be saved in this directory.
- **Topology** (required): neural network model.

  The following network models are currently supported: alexnet_v2, cifarnet, inception_v1, inception_v2, inception_v3, inception_v4, inception_resnet_v2, lenet, resnet_v1_50, resnet_v1_101, resnet_v1_152, resnet_v1_200, resnet_v2_50, resnet_v2_101, resnet_v2_152, resnet_v2_200, vgg_a, vgg_16, vgg_19, mobilenet_v1, mobilenet_v1_025, mobilenet_v1_050, mobilenet_v1_075, nasnet_cifar, nasnet_mobile, nasnet_large
- **Dataset Directory** (required): training dataset path. The dataset example can be downloaded from the link next to this field.
- **Dataset Example**: link for downloading a dataset example.
- **Train Directory** (required): directory that includes the output directory for TensorFlow, summary information, and checkpoints.
- **Batch Size** (required): size of each batch of data imported for training or validation.
- **Learning Rate** (required): The learning rate can be changed due to different learning rate policies. You can change the learning rate policy.
- **Epoch** (required): how many times the data set has been traversed.
  - **Log Cycle**: log output frequency, that is, after how many times of traversal logs will be output for once.
  - **Snapshot Cycle**: snapshot output frequency, that is, after how many times of traversal snapshots will be output for once.
- **Queue** (required): name of the queue on which the job will run. You can only select a queue which you have permission to access. The queue details include:
  - **Queue Status**: **UP** means available.
  - **Available Nodes**: When you put the cursor over it, the number of total nodes and that of free nodes will be displayed.

  - **Available Cores**: When you put the cursor over it, the number of total CPU cores and that of free CPU cores will be displayed.
  - **Available GPU**: If the queue has GPUs, when you put the cursor over it, the number of total GPUs and that of free GPUs will be displayed.
  - **Available Memory**: **UNLIMITED** means no memory limits for jobs.
  - **Wall Time**: **Wall Time** means the maximum execution time of jobs on the queue, while **UNLIMITED** means no limits.
- **Nodes** (required): number of nodes for training.
- **Exclusive**: indicates whether the training can use all the CPU resources. You can change the CPU core setting in non-exclusive mode.
- **GPU Per Node** (required): number of GPUs per node for training.
- **GPU Resource Type**: Click the drop-down list to select the target GPU resource type. Both physial GPU and MIG device can be scheduled. The options can only be displayed when the value of **GPU Per Node** is not less than **1** and more than one GPU type can be selected in the cluster.
- **Wall Time**: maximum execution time of the job. It will be stopped after running for more than the configured **Wall Time**.
- **Optimizer** (required): Different optimizers have different settings.
- **Weight Decay** (required): weight decay ratio.
- **Early Stop**: whether to stop the train job when a monitored metric has stopped improving.
  - **Monitor**: Quantity to be monitored.
  - **Min Delta**: Minimum change in the monitored quantity to qualify as an improvement.
  - **Patience**: Number of worker epochs with no improvement after which training will be stopped.
- **Notify Job Completion**: determines whether to send a notification when the job is completed and the notification method.

  If **Email** is selected, an email will be sent by LiCO once a job is completed.

Step 3. Click **Submit**.

## Submit an Image Classification – AutoDL Train job

Step 1. In the Image Classification area, click **Train → AutoDL Train**.

Step 2. Fill in the required information.

- **Topology** (required): neural network model.

  The following network models are currently supported: Large - Top Accuracy, Big - HPC Server, Standard - Standard Server, Small - Edge Server, Mini - Mobile Device, Tiny - IoT Device.
- **HPO**: hyper-parameter optimization. The options include **None**, **Default**, **BOHB**, and **FAST**. You can change other optimization parameters only if **None** is selected.

For the description of other parameters, see “Submit an Image Classification – Train job” on page 19.

Step 3. Click **Submit**.

## Submit an Object Detection – Train job

Step 1. In the Object Detection area, click **Train → Train**.

Step 2. Fill in the required information.

- **Topology** (required): Neural network models currently supported include Faster R-CNN and YOLO v3.

  **Note:** If the YOLO v3 network model is used, you are advised to set the learning rate to 1e-5 instead of the default value.
- **Pre-trained Model**: A pre-trained model is used to help users improve training performance.
- **Use Random Seed**: indicates whether to shuffle the training data set with random seeds. The default value is **no**.

For the description of other parameters, see "Submit an Image Classification – Train job" on page 19.

Step 3. Click **Submit**.

## Submit an Object Detection – AutoDL Train job

Step 1. In the Object Detection area, click **Train → AutoDL Train**.

Step 2. Fill in the required information.

- **Topology** (required): neural network model.

  The following network models are currently supported: Large - Top Accuracy, Big - HPC Server, Standard - Standard Server, Small - Edge Server, Mini - Mobile Device, Tiny - IoT Device.
- **HPO**: hyper-parameter optimization. The options include **None**, **Default**, **BOHB**, and **FAST**. You can change other optimization parameters only if **None** is selected.

For the description of other parameters, see "Submit an Image Classification – Train job" on page 19.

Step 3. Click **Submit**.

## Submit an Instance Segmentation – Train job

Step 1. In the Instance Segmentation area, click **Train**.

Step 2. Fill in the required information.

**Topology** (required): The **mask-rcnn** neural network model is currently supported.

For the description of other parameters, see "Submit an Image Classification – Train job" on page 19.

Step 3. Click **Submit**.

## Submit a Medical Image Segmentation – Train job

Step 1. In the Medical Image Segmentation area, click **Train**.

Step 2. Fill in the required information.

**Topology** (required): The **unet** neural network model is currently supported.

For the description of other parameters, see "Submit an Image Classification – Train job" on page 19.

**Note:** Medical image segmentation training is available only for datasets of color images.

Step 3. Click **Submit**.

## Submit a Text Classification – Train job

Text classification, also known as text tagging or text categorization, is the process of categorizing text into organized groups. By using Natural Language Processing (NLP), text classifiers can automatically analyze text and then assign a set of pre-defined tags or categories based on its content.

Step 1. In the Text Classification area, click **Train**.

Step 2. Fill in the required information.

- **Topology** (required): The **Bert** neural network model is currently supported.
- **Language** (required): text language used.
- **Max Sequence Length** (required): maximum length of text for training.
- **Lower Case**: determines whether to convert text to the lower case.

For the description of other parameters, see "Submit an Image Classification – Train job" on page 19.

Step 3. Click **Submit**.

## Submit a Seq2seq – Train job

Step 1. In the Seq2seq area, click **Train**.

Step 2. Fill in the required information.

- **Layer Number** (required): indicates how many layers you want.
- **Size of Each Layer** (required): the size of each layer.
- **Vocabulary Size** (required): total vocabulary size you want to translate.

For the description of other parameters, see "Submit an Image Classification – Train job" on page 19.

Step 3. Click **Submit**.

## Submit a Memory Network – Train job

Step 1. In the Memory Network area, click **Train**.

Step 2. Fill in the required information.

- **Feature Size** (required): size of feature in the memory network.
- **Number of Hops** (required): number of hops in the memory network.
- **Embedding Size** (required): size of embedding matrices.
- **Memory Size** (required): maximum size of memory.

For the description of other parameters, see "Submit an Image Classification – Train job" on page 19.

Step 3. Click **Submit**.

## Submit an Image GAN – Train job

Step 1. In the Image GAN area, click **Train**.

Step 2. Fill in the required information.

- **Input Image Height** (required): height of each input image.
- **Input Image Width** (required): width of each input image.
- **Output Image Height** (required): height of each output image.
- **Output Image Width** (required): width of each output image.
- **Samples Output Directory**: the directory which stores the sample pictures for training.

For the description of other parameters, see “Submit an Image Classification – Train job” on page 19.

**Note:** The **Early Stop** item is not supported in the Image Gan train job.

Step 3. Click **Submit**.

## Job submission – Predict

## Submit an Image Classification – Predict job

Step 1. In the Image Classification area, click **Predict → Predict**.

Step 2. Fill in the required information.

- **Job Name** (required): job name. For LiCO 6.1.0 and later versions, the initial job name will be automatically created in the format of "template name + time".
- **Workspace** (required): working directory of the job. Job output files will be saved in this directory.
- **Model Type** (required): the type or predict model. The supported model types include: **Tensorflow Checkpoints**, **TensorRT Model**.
- **Topology** (required): the prediction model. It should be set to the same neural network model as the training job.
- **TensorRT Plan File**: required if **TensorRT Model** is selected. Select the model file (.engine) after exporting TensorRT optimization. **Model Type** specifies the parameters of TensorRT mode.
- **Labels File**: required if **TensorRT Model** is selected. Select the label file (labels.txt) after exporting TensorRT optimization. **Model Type** specifies the parameters of TensorRT mode.
- **Input Directory** (required): the directory contains the images that need to be predicted.
- **Train Directory** (required): the directory contains training checkpoints. **Model Type** specifies the parameters of Tensorflow Checkpoints.
- **Output Directory** (required): the output directory. The prediction result will be saved in this directory.
- **Queue** (required): name of the queue on which the job will run. You can only select a queue which you have permission to access. The queue details include:
  - **Queue Status**: **UP** means available.
  - **Available Nodes**: When you put the cursor over it, the number of total nodes and that of free nodes will be displayed.
  - **Available Cores**: When you put the cursor over it, the number of total CPU cores and that of free CPU cores will be displayed.
  - **Available GPU**: If the queue has GPUs, when you put the cursor over it, the number of total GPUs and that of free GPUs will be displayed.
  - **Available Memory**: **UNLIMITED** means no memory limits for jobs.

  - **Wall Time**: the maximum execution time of jobs on the queue, while **UNLIMITED** means no limits.
- **Nodes** (required): number of nodes for training.
- **Exclusive** (required): indicates whether the training can use all the CPU resources. You can change the CPU core setting in non-exclusive mode.
- **GPU Per Node** (required): number of GPUs per node for training.
- **GPU Resource Type**: Click the drop-down list to select the target GPU resource type. Both physial GPU and MIG device can be scheduled. The options can only be displayed when the value of **GPU Per Node** is not less than **1** and more than one GPU type can be selected in the cluster.
- **Wall Time**: maximum execution time of the job. It will be stopped after running for more than the configured **Wall Time**.
- **Notify Job Completion**: determines whether to send a notification when the job is completed and the notification method.

Step 3. Click **Submit**.

## Submit an Image Classification – AutoDL Predict job

Step 1. In the Image Classification area, click **Predict → AutoDL Predict**.

Step 2. Fill in the required information.

For the parameter description, see “Submit an Image Classification – Predict job” on page 23.

Step 3. Click **Submit**.

## Submit an Object Detection – Predict job

Step 1. In the Object Detection area, click **Predict → Predict**.

Step 2. Fill in the required information.

For the parameter description, see “Submit an Image Classification – Predict job” on page 23.

Step 3. Click **Submit**.

## Submit an Object Detection – AutoDL Predict job

Step 1. In the Object Detection area, click **Predict → AutoDL Predict**.

Step 2. Fill in the required information.

For the parameter description, see “Submit an Image Classification – Predict job” on page 23.

Step 3. Click **Submit**.

## Submit an Instance Segmentation – Predict job

Step 1. In the Instance Segmentation area, click **Predict**.

Step 2. Fill in the required information.

For the parameter description, see “Submit an Image Classification – Predict job” on page 23.

Step 3. Click **Submit**.

## Submit a Medical Image Segmentation – Predict job

Step 1. In the Medical Image Segmentation area, click **Predict**.

Step 2. Fill in the required information.

For the parameter description, see "Submit an Image Classification – Predict job" on page 23.

**Note:** Medical image segmentation prediction is available only for color images.

Step 3. Click **Submit**.

## Submit a Text Classification – Predict job

Step 1. In the Text Classification area, click **Predict**.

Step 2. Fill in the required information.

- **Topology** (required): The **Bert** neural network model is currently supported.
- **Language** (required): text language used.
- **Max Sequence Length** (required): maximum length of text for training.
- **Lower Case**: determines whether to convert text to the lower case.
- **Input File** (required): the TSV file that needs to be translated.

For the description of other parameters, see "Submit an Image Classification – Predict job" on page 23.

Step 3. Click **Submit**.

## Submit a Seq2seq – Predict job

Step 1. In the Seq2seq area, click **Predict**.

Step 2. Fill in the required information.

- **Input File** (required): the TXT file that needs to be translated. Its format must be the same as the provided TXT file in the **Train** job you submitted.
- **Layer Number** (required): specifies how many layers you want.
- **Size of Each Layer** (required): specifies the size of each layer.
- **Vocabulary Size** (required): total vocabulary size you want to translate.

For the description of other parameters, see "Submit an Image Classification – Predict job" on page 23.

**Note:** Apart from **Input File**, the parameter values must be exactly the same as those in the **Train** job.

Step 3. Click **Submit**.

## Submit a Memory Network – Predict job

Step 1. In the Memory Network area, click **Predict**.

Step 2. Fill in the required information.

- **Input File** (required): the TXT file that needs Q&A. Its format must be the same as the provided TXT file in the **Train** job you submitted. In addition, the question must end with two separate tabs.
- **Feature Size** (required): size of feature in the memory network.

- **Number of Hops** (required): number of hops in the memory network.
- **Embedding Size** (required): size of embedding matrices.
- **Memory Size** (required): maximum size of memory.

For the description of other parameters, see “Submit an Image Classification – Predict job” on page 23.

Step 3. Click **Submit**.

## Submit an Image GAN – Predict job

Step 1. In the Image GAN area, click **Predict**.

Step 2. Fill in the required information.

- **Input Image Height** (required): height of each input image.
- **Input Image Width** (required): width of each input image.
- **Output Image Height** (required): height of each output image.
- **Output Image Width** (required): width of each output image.
- **Batch Size** (required): number of the final generated images. This number must have a square root.
- **Output Directory**: the directory which stores the result pictures of prediction.

For the description of other parameters, see “Submit an Image Classification – Predict job” on page 23.

**Note:** The parameters listed above must have exactly the same values as those in the **Train** job you submitted.

Step 3. Click **Submit**.

# Job submission – Export

## Submit an Image Classification – Export job

Step 1. In the **Image Classification** are, click **Export**.

Step 2. Fill in the following fields:

- **Job Name** (required): job name. For LiCO 6.1.0 and later versions, the initial job name will be automatically created in the format of "template name + time".
- **Workspace** (required): working directory of the job. Job output files will be saved in this directory.
- **Topology** (required): neural network model.

  The following network models are currently supported: alexnet_v2, cifarnet, inception_v1, inception_v2, inception_v3, inception_v4, inception_resnet_v2, lenet, resnet_v1_50, resnet_v1_101, resnet_v1_152, resnet_v1_200, resnet_v2_50, resnet_v2_101, resnet_v2_152, resnet_v2_200, vgg_a, vgg_16, vgg_19, mobilenet_v1, mobilenet_v1_025, mobilenet_v1_050, mobilenet_v1_075, nasnet_cifar, nasnet_mobile, and nasnet_large.
- **Train Directory** (required): Include the export directories of TensorFlow, abstract, and checkpoint.
- **Max Batch Size** (required): The size of each batch of imported data for training or verifying.
- **Target** (required): Export format. The supported types include: TensorRT-FP32, TensorRT-INT8.

  - **Calibration Dataset**: Required if **TensorRT-INT8** is selected. This directory stores multiple train images for improving accuracy.
  - **Calibration Batch Size**: Required if **TensorRT-INT8** is selected. The batch size for the inner calibration job within exporting.
- **Export Directory** (required): The directory saving the exported data.
- **Queue** (required): name of the queue on which the job will run. You can only select a queue which you have permission to access. The queue details include:
  - **Queue Status**: **UP** means available.
  - **Available Nodes**: When you put the cursor over it, the number of total nodes and that of free nodes will be displayed.
  - **Available Cores**: When you put the cursor over it, the number of total CPU cores and that of free CPU cores will be displayed.
  - **Available GPU**: If the queue has GPUs, when you put the cursor over it, the number of total GPUs and that of free GPUs will be displayed.
  - **Available Memory**: **UNLIMITED** means no memory limits for jobs.
  - **Wall Time**: **Wall Time** means the maximum execution time of jobs on the queue, while **UNLIMITED** means no limits.
- **Exclusive**: indicates whether the training can use all the CPU resources. You can change the CPU core setting in non-exclusive mode.
- **CPU Cores Per Node**: Required if **Exclusive** is not selected. The number of available CPU cores in each train node.
- **GPU Per Node** (required): number of GPUs per node for training.
- **Wall Time**: maximum execution time of the job. It will be stopped after running for more than the configured **Wall Time**.
- **Notify Job Completion**: determines whether to send a notification when the job is completed and the notification method.

  If **Email** is selected, an email will be sent by LiCO once a job is completed.

Step 3. Click **Submit**.

# Deployment

Using a trained model in your production environment is a very customizing process. This section only provides how to do model inference outside the LiCO environment. You can modify the inference code based on this Guide and merge it into your environment.

## Install LeTrain

- Your deployment environment is CentOS/RHEL 8.2.0 or SLES 15.2.
- TensorFlow V1.9.0 has been installed.

Step 1. Obtain the LeTrain release package from https://hpc.lenovo.com/lico/downloads/6.1/letrain-v1.3.1.tar.gz.

Step 2. Copy this package to `/opt` in your environment.

Step 3. Run the following command to install LeTrain:

```
apt-get install -y python-setuptools libsm-dev python-tk

cd /opt
```

```
tar -xvf letrain-v1.3.1.tar.gz

chmod -R 777 letrain-v1.3.1

ln -s /opt/letrain-v1.3.1 /opt/letrain

pip install setuptools==39.1.0 --force-reinstall

pip install -r /opt/letrain/requirements.txt

pip install -r /opt/letrain/applications/maskrcnn/requirements.txt

cd /opt/letrain/applications/frcnn/lib

make
```

## Export the trained model

The trained model is stored in the "Train Directory" you selected for the training job.

Step 1. Click the system tools icon  in the upper-right corner of the home page.

Step 2. Find the train directory that contains the trained model, compress it, and then download it.

Step 3. Upload the compressed directory to your deployment environment, and then decompress it.

## Run the inference toolkit code

All the inference toolkit code is open-sourced. You can find the source code in `/opt/letrain/inference/`.

### Image Classification

**Code:**

`classification/class_inference.py`

**Parameters:**

**input_dir**: the directory that stores the images to be processed. The images should be stored directly under this directory.

**model_name**: name of model, which must be the same as your selection during training

**checkpoint_dir**: the directory that stores the content of "Train Directory"

**output_dir**: the output directory

### Object Detection

**Code:**

`frcnn/frcnn_inference.py`

**Parameters:**

**model_name**: name of the model, which must be the same as the one selected for the training job.

**input_dir**: the directory that stores the images to be processed. The images should be stored directly under this directory.

**checkpoint_dir**: the directory that stores the content of the training directory

**category_file**: the same category file's absolute path as in the training dataset

**output_dir**: the output directory

## Instance Segmentation

**Code:**

`maskrcnn/maskrcnn_inference.py`

**Parameters:**

**input_dir**: the directory that stores the images to be processed. The images should be stored directly under this directory.

**checkpoint_dir**: the directory that stores the content of "Train Directory"

**category_file**: the instance_train.json file in "Train Directory"

**output_dir**: the output directory

## Medical Image Segmentation

**Code:**

`unet/unet_inference.py`

**Parameters:**

**input_dir**: the directory that stores the images to be processed

**input_fname_pattern**: the filter of image files, which can be *.jpg, *.png, *.bmp, or *.jpeg

**checkpoint_dir**: the directory that stores the content of "Train Directory"

**output_dir**: the output directory

## Seq2Seq

**Code:**

`translate/translate_inference.py`

**Parameters:**

**num_layers**: number of layers, which is defined during training

**size**: size of each layer, which is defined during training

**checkpoint_dir**: the directory that stores the content of "Train Directory"

**from_vocab_size**: vocabulary size of the source language

**to_vocab_size**: vocabulary size of the target language

**input_file**: the file containing the sentences to be translated

**output_dir**: the output directory

## Memory Network

### Code:

`kvmem/kvmem_inference.py`

### Parameters:

**input_file**: the TXT file that needs Q&A. Its format must be the same as the provided TXT file in the **Train** job you submitted. In addition, the question must end with two separate tabs.

**checkpoint_dir**: the directory that stores the content of the training directory

**output_dir**: the output directory

## Image GAN

### Code:

`dcgan/dcgan_inference.py`

### Parameters:

**input_height**: height of images set during training

**input_width**: width of images set during training

**output_height**: height of output images

**output_width**: width of output images

**checkpoint_dir**: the directory that stores the content of "Train Directory"

**output_dir**: the output directory

# Chapter 4. AI Studio

AI Studio is an E2E feature that delivers Data Scientist abilities from creating datasets to deploying trained models as services. It is integrated with not only improved multi-node scheduling methods but also the well-refined models from Lenovo Research.

By using **AI Studio → Datasets**, users can create a new dataset or a reference to an existing dataset, as well as add labels, select new objects, and make new segmentation on images within a dataset.

By using **AI Studio → Training Tasks**, users can start training easily with the datasets created before and control the training process in a fine granularity. Users can also start a tuning task, perform a quick test on the output, and find out the best model.

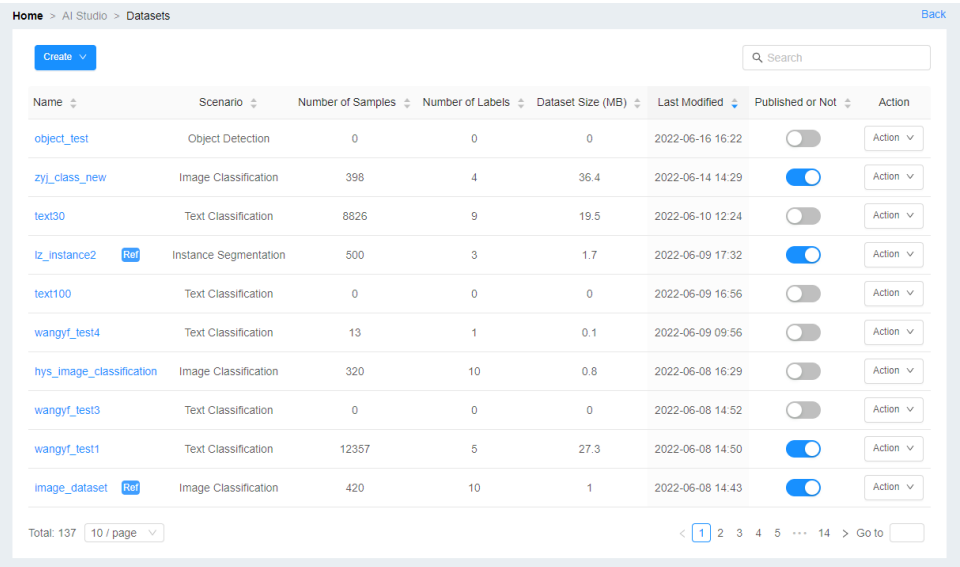
By using **AI Studio → Trained Models**, users can easily test multiple models from the training process, find out the best to be published to a remote repository, and quickly deploy the models for public usage or testing.

Besides all of the functions above, AI Studio provides public APIs for users to integrate all its functions into their automated workflows, such as Jenkins and Blue Ocean.

## Datasets

Select **AI Studio → Datasets** from the left navigation pane.

The Datasets page is displayed.




## Dataset information

The dataset information is described as follows:

- **Name**: dataset name
  - Without the **Ref** flag: The dataset is created as a new dataset.

– With the **Ref** flag: The dataset is created as a reference dataset. A reference dataset is just a reference to a local dataset.

  **Note:** Reference datasets are set to the published state by default, and **Edit** and **Duplicate** operations are unavailable.

- **Scenario**: dataset type, which can be **Image Classification**, **Object Detection**, **Instance Segmentation**, or **Text Classification**.
- **Published**: Only unpublished datasets can be edited and deleted, and only published datasets can be used for creating training tasks.

## Dataset operations

The following dataset operations are available:

**Action**

- **Edit**: Edit the dataset information. This action is available only for a new and unpublished dataset.
- **Duplicate**: Copy all the data from the current dataset to a new one. This action is available only for a new dataset.
- **Delete**: Delete the dataset.

## Dataset details

Click the name of the dataset for which you want to view its details.

**Action**: Available actions on the dataset are the same as those on the Datasets page.

**Notes:**

- Different color blocks represent different kinds of labels, and the size of a color block depends on the proportion of images with the corresponding label.
- When you hover over a color block, the number of images with the corresponding label will be displayed.
- For reference datasets, the **Folder Path** information will be displayed.

## Create a dataset

LiCO allows you to create a new dataset or a reference dataset.

### Create a new dataset

Step 1. Click **Create** in the upper left corner and select **Create a New Dataset**.

Step 2. Fill in the required information.

- **Dataset Name** (required): Enter 3 to 50 characters not beginning or ending with special characters. Only letters, digits, underscores (_), and Chinese characters are allowed.
- **Scenario** (required): Select a type for the dataset to be created.
- **Language**: required when creating a text classification dataset. The default value is English.

Step 3. Click **OK**.

### Create a reference dataset

Step 1. Click **Create** in the upper left corner and select **Create a Reference Dataset**.

Step 2. Fill in the required information.

- **Dataset Name** (required): Enter 3 to 50 characters not beginning or ending with special characters. Only letters, digits, underscores (_), and Chinese characters are allowed.
- **Scenario** (required): Select a type for the dataset to be created.
- **Folder Path** (required): Select the folder where the dataset to be referenced is located.
- **Language**: required when creating a text classification dataset. The default value is English.
- **Dataset Analysis**: Dataset analysis takes several minutes, depending on the size of the dataset.

Step 3. Click **OK**.

**Notes:**

- After you click **OK**, the dataset formation will be validated to get metadata of this dataset. This may take a while when the size of the existing dataset is very large. You can select a scenario and click **Dataset Example** for formation details about the selected scenario.
- A reference dataset successfully created is set to the published state by default.

## Edit an image classification dataset

Step 1. On the Datasets page, find the target unpublished image classification dataset, and then select **Action → Edit**.

Step 2. Perform the required operation on the dataset.

The following operations are supported on this page:

- **Create Category**: Create a new category for the dataset.
- **Action**: Perform an action on the selected category.
  - **Rename**: Rename the category.
  - **Delete**: Delete the category.
  - **Edit**: Switch to the category image management page.

### Category image operations

On the category image management page, the following operations are available for images:

#### Upload

Upload images to the current dataset.

- **Upload File**: Upload a single image or a ZIP, TAR, or GZIP package.
  1. Click **Action → Upload → Upload File** in the upper left corner.

     The Select File dialog is displayed.
  2. Select a single image or a package.
  3. Click **Select File**.

     The selected image or package is uploaded to the current category.
- **Upload Folder**: Upload a folder.
  1. Click **Action → Upload → Upload Folder** in the upper left corner.

     The Select Folder dialog is displayed.
  2. Select a folder.
  3. Click **Select Folder**.

The selected images are uploaded to the current category.

**Operations in the thumbnail pane and preview pane**

-  : Zoom in the current image.
-  : Zoom out the current image.
-  : Scroll up to see the thumbnail.
- : Scroll down to see the thumbnail.
-  : Get help information.

**Delete**: Delete the current image.

**Move Image**: Move the current image to another category.

**Back**: Return to the page for editing the image classification dataset.

## Edit an object detection or instance segmentation dataset

On the Datasets page, find the unpublished object detection or unpublished instance segmentation dataset you want to edit, and then select **Action → Edit**.

An annotation page is displayed. This page is similar to the category image management page but provides more functions.

### Image operations

On the annotation page, the following operations are available for images:

**Upload**: Upload images to the current dataset. Refer to “Upload” on page 33.

**Import**: Import CVAT dataset to the target dataset.

- Import COCO dataset for instance segmentation:
  1. Select **Action → Import → COCO 1.0** from the left navigaiton pane. The Select File page is displayed.
  2. On the Select File page, select the target ZIP file.
  3. Click **Select File** on the upper tool bar. The target ZIP file will be uploaded to the target dataset.
- Import VOC dataset for object detection:
  1. Select **Action → Import → PASCAL VOC1.1** from the left navigaiton pane. The Select File page is displayed.
  2. On the Select File page, select the target ZIP file.
  3. Click **Select File** on the upper tool bar. The target ZIP file will be uploaded to the target dataset.

**All**: All images are displayed.

**Unlabeled**: Only the unannotated images are displayed.

**Labeled**: Only the annotated images are displayed.

**Delete**: Delete the current image and its annotation information.

**Operations in the thumbnail pane and preview pane**: Refer to “Operations in the thumbnail pane and preview pane” on page 34.

## Annotation operations

On the annotation page, the following operations are available for annotations:

### Operations in the preview pane

- : Return to the previous annotation operation.
- : Restore the cancelled annotation operation.

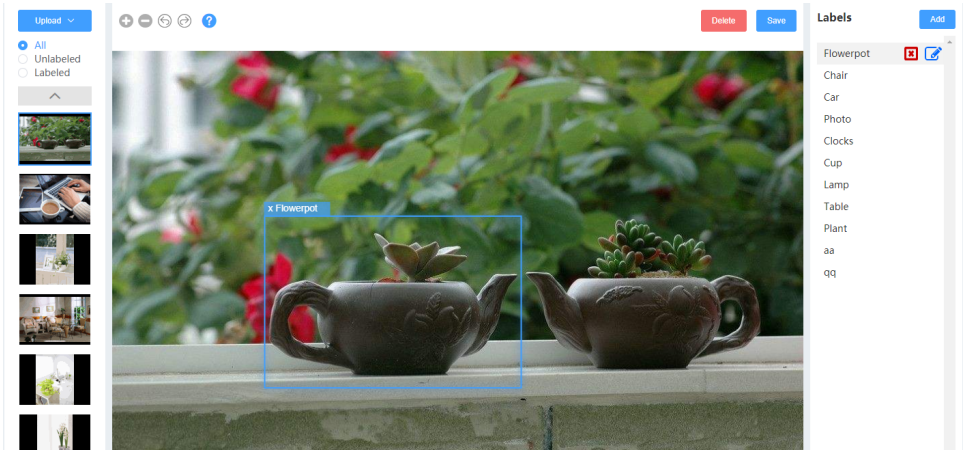
**Note:** The preceding two functions are available only when the annotation information has not been saved.

- **x**: Delete the current annotation. **x** is in the upper left corner of the annotation box.
- **Save**: Save the annotation information for the current image.

### Add an annotation (for object detection datasets)

1. Select the desired label.
2. Select the image to which the annotation is to be added.
3. Hold down the left mouse button and drag.

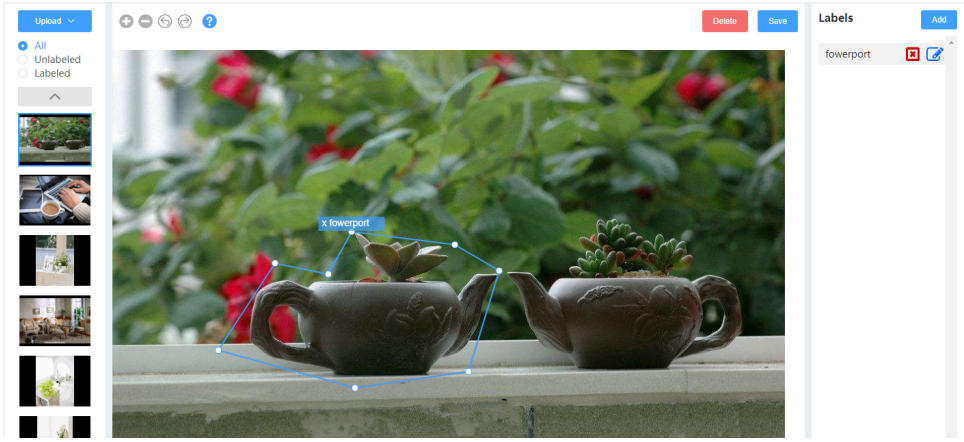
   The label is added successfully, as shown below.




### Add an annotation (for instance segmentation datasets)

1. Select the desired label.
2. Select the image to which the annotation is to be added.
3. Hold down the **Alt** key and left mouse button to draw a closed shape.

   The label is added successfully, as shown below.




**Notes:**

- The annotation must be a closed graph. Otherwise, the annotation cannot be completed.
- When drawing a closed shape, you can right-click to undo the last drawn line.
- Firefox is not recommended, because of its shortcut conflicts with **Alt**. Google Chrome is recommended.

### Label operations

On the annotation page, the following operations are available for labels:

- **Add**: Add a new label.
-  : Rename the selected label.
-  : Delete the selected label.

  **Note:** When a label is deleted, the annotation on the image will also be deleted.

## Edit a text classification dataset

Step 1. On the Datasets page, find the target unpublished text classification dataset, and then select **Action → Edit**.

Step 2. Perform the required operation on the dataset.
The following operations are supported on this page:

- To add a new text classification dataset:
  1. Click **Add**. The Add Sample page is displayed.
  2. On the Add Sample page, input the category and sample.
  3. Do one of the following:
     - To cancel adding the new text classification dataset, click **Cancel**.
     - To submit a new text classification dataset, click **Submit**.
     - To Submit a new text classification dataset and add the next new one, click **Submit and Continue**, and repeat the above steps.
- To import data file:
  1. Click **Import**. The Import Sample page is displayed.
  2. On the Import Sample page, click **Browse** and select the target TSV file.

**Note:** Users can click **Data File Example** to download and view the example.

3. Click **Cancel** or **Submit** to cancel or submit the data file.

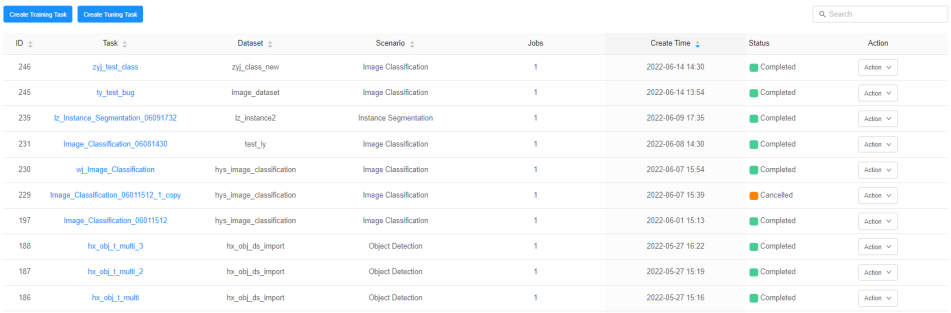
- To delete the text classification dataset:
  - Select one or more target datasets from the list and click **Delete**.
  - Find the target dataset and click **Action → Delete** on the right.
- To edit the target text classification dataset:
  1. Find the target dataset, click **Action → Edit** on the right. The Edit Sample page is displayed.
  2. On the Edit Sample page, modify the category and sample.
  3. Click **Cancel** or **OK** to cancel or submit the modification.

## Training tasks

Training tasks are divided into training and tuning tasks. To create multi-node tasks for high-speed training, choose training tasks. To create multiple single-node tasks at a time, choose tuning tasks. It is recommended that you use one or more tuning tasks to find a better combination of hyper-parameters before starting a full training task. This is one of the best practices when you are faced with a training task in an unknown area.

Select **AI Studio → Training Tasks** from the left navigation pane.

The Training Tasks page is displayed.




The parameters on the Training Tasks page are described as follows:

- **Task**: task name
- **Dataset**: dataset used
- **Scenario**: dataset type
- **Jobs**: number of jobs in a task
- **Status**: task status, which is defined according to the following rules:
  - **Running**: At least one job in the task is running.
  - **Cancelled**: All the jobs in the task are cancelled, or the task is cancelled by the user.
  - **Completed**: All the jobs in the task are completed.
  - **Waiting**: All the jobs in the task are waiting to run.
  - **Failed**: All the jobs in the task are failed.
- **Action**: action on the task, which can be:
  - **Cancel**: Only running and waiting tasks can be cancelled. Once a task is cancelled, all of its jobs will be cancelled.

– **Delete**: Only cancelled, completed, and failed tasks can be deleted. Once a task is deleted, all of its jobs will also be deleted.

You can also perform the following operations on the Training Tasks page:

- Sort tasks: Sort tasks by **ID**, **Task**, **Dataset**, **Scenario**, **Jobs**, and **Create Time** through clicking the corresponding column name in the task list.
- Search for tasks: Enter a task name or dataset name in the search bar and click **Enter**. The search result will be displayed.

# Submit a task

## Submit a training task

Step 1. Click **Create Training Task** in the upper left corner.

Step 2. Fill in the required information on the **Dataset** tab.

The parameters on the **Dataset** tab are described as follows:

- **Task Name**: task name. For LiCO 6.1.0 and later versions, the initial task name will be automatically created in the format of "template name + time".
- **Dataset**: Select a dataset for the task. Only published datasets can be chosen, and unpublished datasets will be shown in grey.
- **Training Ratio**: ratio of training data in the selected dataset. The remaining data will be used for validation. For reference datasets, the value of **Training Ratio** cannot be changed.

Step 3. Click **Next**.

Step 4. Fill in the following required information: **Mode**: select the task mode. If the task dataset is the image-based dataset or new object detection dataset, you can select **Standard** or **AutoDL**. For other types of dataset, this field will not be displayed.

- If **Standard** is selected or if there is no **Mode** parameter, fill in the following required information:
  – **Topology**: neural network model. Topologies supported for image classification, object detection, instance segmentation, and text classification are listed in the following table.

*Table 1. Topologies applicable for different scenarios*

| Scenario | Available Topologies |
|---|---|
| Image classification | alexnet_v2, cifarnet, inception_v1, inception_v2, inception_v3, inception_v4, inception_resnet_v2, lenet, resnet_v1_50, resnet_v1_101, resnet_v1_152, resnet_v1_200, resnet_v2_50, resnet_v2_101, resnet_v2_152, resnet_v2_200, vgg_a, vgg_16, vgg_19, mobilenet_v1, mobilenet_v1_025, mobilenet_v1_050, mobilenet_v1_075, nasnet_cifar, nasnet_mobile, nasnet_large |
| Object detection | Faster R-CNN, YOLO v3 |
| Instance segmentation | mask-rcnn |
| Text classification | Bert |

  – **Max Sequence Length**: the maximum sequence length of training task. This option will be displayed when **Bert** is selected in the **Topology** drop-down list.
  – **Lower Case**: switch the text to lower case. This option will be displayed when **Bert** is selected in the **Topology** drop-down list.

  - **Batch Size**: size of each batch of data imported for training or validation.
  - **Learning Rate**: Click the value box, and then set the learning rate policy as required in the displayed Learning Rate Settings dialog.
  - **Optimizer**: Click the value box, and then specify the optimizer settings in the displayed Optimizer Setting dialog.
  - **Weight Decay**: weight decay ratio.
- If **AutoDL** is selected, fill in the following required information:
  - **Topology**: neural network model. It supports Large - Top Accuracy, Big - HPC Server, Standard - Standard Server, Small - Edge Server, Mini - Mobile Device, and Tiny - IoT Device.
  - **HPO**: hyper-parameter optimization. The options include **None**, **Default**, **BOHB**, and **FAST**. You can change other optimization parameters only if **None** is selected.

Step 5. Click **Next**.

Fill in the required information.

- **Epoch** (required): Number of times that the dataset has been traversed.
  - **Log Cycle**: log output frequency, the number of times that traversal logs will be output for once.
  - **Snapshot Cycle**: snapshot output frequency, the number of times that traversal snapshots will be output for once.
- **Early Stop**: whether to stop training when a monitored metric has stopped improving.
  - **Monitor**: Quantity to be monitored.
  - **Min Delta**: Minimum change in the monitored quantity to qualify as an improvement.
  - **Patience**: Number of worker epochs with no improvement after which training will be stopped.
- **Queue**: name of the queue on which the job will run. You can only select a queue which you have permission to access.
- **Nodes**: number of nodes for training.
- **Exclusive**: indicates whether the training can use all the CPU resources.
- **CPU Cores Per Node**: number of CPU cores per node for training. This parameter is displayed only when the **Exclusive** check box is not selected.
- **GPU Per Node**: number of GPUs per node for training. At least one GPU should be configured for each node.
- **GPU Resource Type**: Click the drop-down list to select the target GPU resource type. Both physial GPU and MIG device can be scheduled. The options can only be displayed when the value of **GPU Per Node** is no less than **1** and more than one GPU type can be selected in the cluster.

Step 6. Click **Next**.
The **Confirm** tab is displayed, which lists some important information about this task. You can set the method to send a notification once the job is completed.

**Note:** To change the parameter settings, click **Previous** to return to the previous tab.

Step 7. Click **Submit** to submit the training task.

## Submit a tuning task

A tuning task is similar to a training task, except that a tuning task supports multiple jobs.

Step 1. Click **Create Tuning Task** in the upper left corner.

Step 2. Fill in the required information on the **Dataset** tab.

The parameters on the **Dataset** tab are described as follows:

- **Task Name**: task name. For LiCO 6.1.0 and later versions, the initial task name will be automatically created in the format of "template name + time".
- **Dataset**: Select a dataset for the task. Only published datasets can be chosen, and unpublished datasets will be shown in grey.
- **Training Ratio**: ratio of training data in the selected dataset. The remaining data will be used for validation. For reference datasets, the value of **Training Ratio** cannot be changed.

Step 3. Click **Next**.

Fill in the required information.

- **Topology**: neural network model. Available scenarios and topologies are the same as those for the training task. Refer to Table 1 “Topologies applicable for different scenarios” on page 38.
- **Max Sequence Length**: the maximum sequence length of training task. This option will be displayed when **Bert** is selected in the **Topology** drop-down list.
- **Lower Case**: switch the text to lower case. This option will be displayed when **Bert** is selected in the **Topology** drop-down list.
- **Batch Size**: size of each batch of data imported for training or validation.
- **Learning Rate**: learning rate policy.
- **Optimizer**: optimizer settings.
- **Weight Decay**: weight decay ratio.
- **Pre-trained Model**: A pre-trained model is used to help users improve training performance.

**Notes:** Except for **Topology** and **Pre-trained Model**, all the other parameters support particular settings and multiple values.

- To configure the particular settings for a parameter, click the value box of the parameter.
- To add a value for a parameter, click  in the same line as the parameter.
- The number of parameter values determines the number of jobs created for the tuning task. For example, if **Batch Size** has two values, **Learning Rate** has two values, **Optimizer** has three values, **Weight Decay** has two values, the number of jobs created for the tuning task will be 2 x 2 x 3 x 2 = 24, with each job configured with different parameter values.

Step 4. Click **Next**.

Step 5. Fill in the required information.

For the parameter description, see “Submit a training task” on page 38.

All these resource settings on the **Running** tab are configured for a single job, not for the whole task. Each job runs only on one node while the whole task runs on multiple nodes.

Step 6. Click **Next**.
The **Confirm** tab is displayed, which lists some important information about this task.

**Note:** To change the parameter settings, click **Previous** to return to the previous tab.

Step 7. Click **Submit** to submit the tuning task.

**Note:** A maximum of 99 jobs can be created for a tuning task. If there are more than 99 jobs, the task will fail to be submitted.

# View job information of a task

## Job comparison

Step 1. Click the name of a task with multiple jobs.
The **Job Comparison** tab is displayed.

Step 2. In the job list in the left pane, select required jobs to make comparison.

Available dimensions for comparison are as follows:

- **Epoch**: Compare the job loss values with epoch being the X-axis.
- **Duration**: Compare the job loss values with time being the X-axis.

The loss values of the selected jobs are displayed in the line charts on the right.

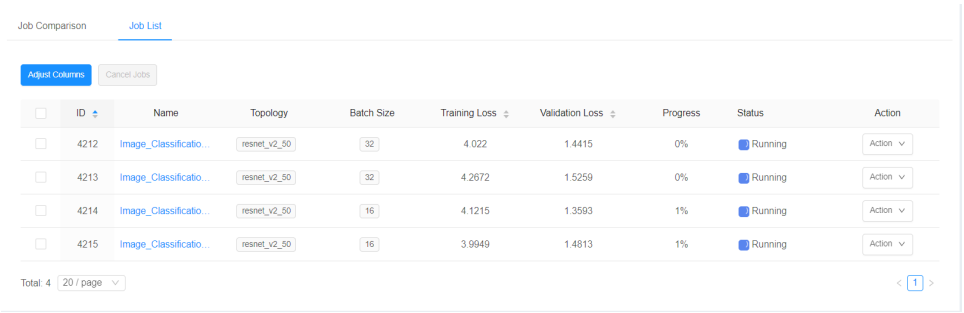
**Notes:** Tips for you to perform job comparison are as follows:

- Click the blue arrow on the right side of the job list to show or hide the job list.
- Click the black arrow in the same line as a job to show or hide its parameters.

## Job list

Click **Job List** next to **Job Comparison**.

The **Job List** tab is displayed.




The parameters on the **Job List** tab are described as follows:

- **Name**: job name
- **Topology**: neural network model. (You can decide whether to display this parameter by using the **Adjust Parameters** function above the job list.)
- **Batch Size**: size of each batch of data imported for training or validation. (You can decide whether to display this parameter by using the **Adjust Parameters** function above the job list.)
- **Training Loss**: training loss value
- **Validation Loss**: validation loss value
- **Progress**: training progress of the job, which is obtained by dividing the current step by the largest step.
- **Status**: job status
- **Action**: action on the job
  - **Cancel**: This operation is available only for running and queuing jobs.

- **Copy**: After you click **Copy**, a page for creating a training task is displayed. All the parameters have been already configured in the same way as the selected job. Follow the steps to finish the submission.
- **Save Model**: This operation is available only for cancelled and completed jobs. Click **Save Model**, enter the model name, and then click **OK** to save the last checkpoint in the training result of the selected job as a model. You can get details of this model on the Trained Models page.

## Adjust parameters

The third and fourth parameters in the job list are default parameters to be displayed. To change the parameters to be displayed, use the **Adjust Parameters** function.

Step 1. Click **Adjust Columns** above the job list.
The Select Parameters dialog is displayed.

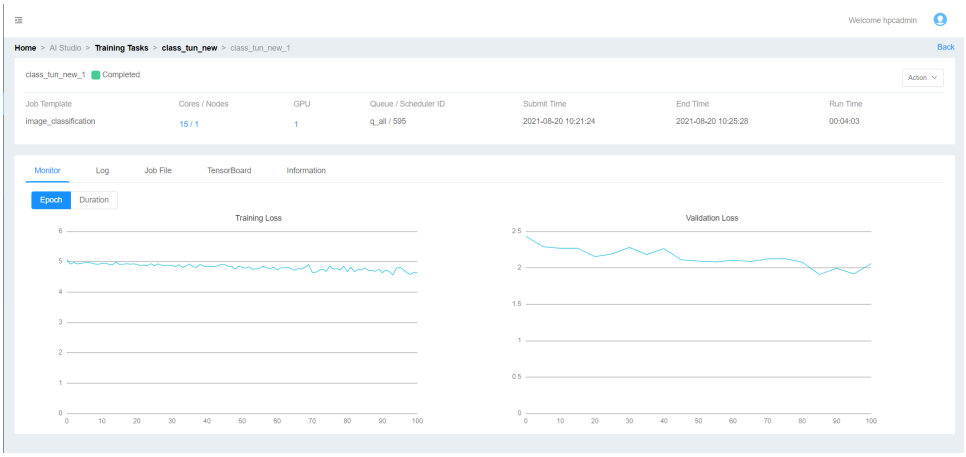
Step 2. Select two parameters to be displayed.

Step 3. Click **OK**.
The selected two parameters are displayed in the third and fourth columns of the job list.

## View details of a single job

For a task with only one job, click the task name; for a task with multiple jobs, click the task name and then click a job name on the **Job List** tab.

The Job details page is displayed.




The parameters and tabs on the job details page are described as follows:

- **Action**
  - **Browse**: After you click **Browse**, the File Manager window is displayed. The path located is the job workspace.
  - **Cancel**: This operation is available only for running, queuing, and waiting jobs.
  - **Copy**: Copy the current job to create a training task.
  - **Save Model**: Save the training result of the current job as a model.
- **Monitor**
  - **Epoch**: number of epochs performed by a task
    - **Training Loss** chart: depicts training loss, with epoch being the X-axis
    - **Validation Loss** chart: depicts training loss, with epoch being the X-axis

  - **Validation mAP** chart (only supported in Object Detection AutoDL): depicts validation meanAP, with epoch being the X-axis
- **Duration**: task execution duration
  - **Training Loss** chart: depicts training loss, with task running duration being the X-axis
  - **Validation Loss** chart: depicts training loss, with task running duration being the X-axis
  - **Validation mAP** chart (only supported in Object Detection AutoDL): depicts validation meanAP, with epoch being the X-axis

- **Log**

  Provides the output file for this job, including the file path and file content.

- **Job File**

  Provides the slurm file for this job, including the file path and file content.

- **Information**

  Provides the information about this job.

- **TensorBoard**

  LiCO has built-in TensorBoard features for TensorFlow programs, helping users monitor the running process of TensorFlow programs on the graphical interface. On the **TensorBoard** tab, you can do the following:

  - Click **View**: Launch TensorBoard.
  - Click **Refresh**: Refresh TensorBoard.
  - Click **New Window** to display a complete view.

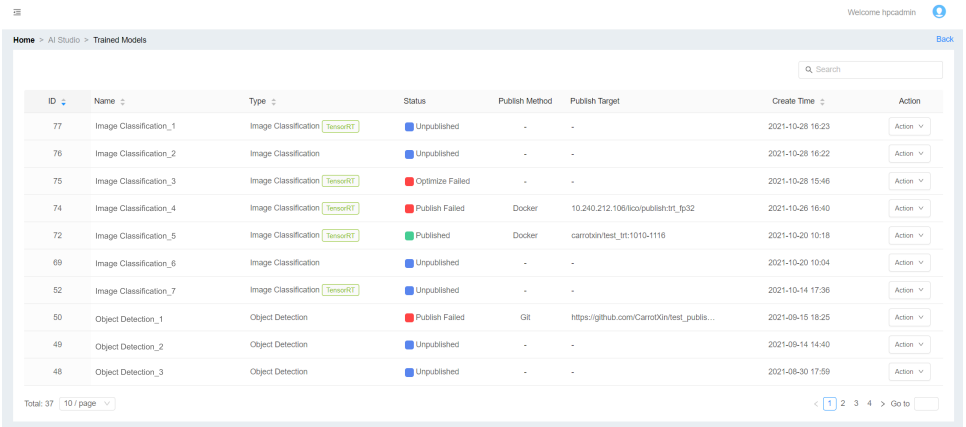
# Trained models

This section describes how to publish a trained model or deploy it as a service.

## View models

Select **AI Studio → Trained Models** from the left navigation pane.

The Trained Models page is displayed.



This page shows details of all the trained models, and provides some convenient functions such as sorting models by field and searching for models by keyword. It also supports multiple actions on the trained models. Different states of trained models are described as follows:

- **Optimizing**: The model is being optimized.
- **Optimize Canceling**: The optimization task is being cancelled.
- **Optimize Failed**: The optimization task is failed.
- **Unpublished**: The trained model is unpublished.
- **Publishing**: The trained model is being published.
- **Published**: The trained model is published successfully.
- **Publish Failed**: The trained model failed to be published.

## Delete a model

Step 1. Select **AI Studio → Trained Models** from the left navigation pane.

Step 2. Find the model you want to delete, and then select **Action → Delete**.

Step 3. Click **OK**.

## Publish a model

Step 1. Select **AI Studio → Trained Models** from the left navigation pane.

Step 2. Find the model you want to publish, and then select **Action → Publish**. Select **Git** or **Docker** as required.

Step 3. Fill in the required information. Refer to “Create a Git publishing task” on page 16 or “Create a Docker publishing task” on page 17.

Step 4. Optional: After the model is published, click **Action → View Publishing** in the same row as the model on the Trained Models page.
Logs and information of the publishing process are displayed.

Step 5. Optional: If you use Docker for publishing, refer to “Deploying a publishing image” on page 93 ot use the image.

## Republish a model

**Note: Republish** action is available only for models in **Published** state.

Step 1. Select **AI Studio → Trained Models** from the left navigation pane.

Step 2. Find the model you want to republish, and then select **Action → Republish**.

Step 3. Click **OK**.

## Stop a publishing model

Step 1. Select **AI Studio → Trained Models** from the left navigation pane.

Step 2. Find the publishing model you want to stop, and then select **Action → Stop**.

Step 3. Click **OK**.

## Optimize a model

Step 1. Select **AI Studio → Trained Models** from the left navigation pane.

Step 2. Find the target model and click **Action → Optimize**.

**Note:** Only non-AutoDL Image Classification model can be optimized.

Step 3. Fill in the required fields.

Step 4. Click **OK**.

## Stop optimizing a model

Step 1. Select **AI Studio → Trained Models** from the left navigation pane.

Step 2. Find the target model and click **Action → Stop**.

Step 3. Click **OK**.

## Deploy a model

Step 1. Select **AI Studio → Trained Models** from the left navigation pane.

Step 2. Find the model you want to deploy, and then select **Action → Deploy**.

Step 3. Fill in the required information.

Step 4. Click **OK** to deploy the model as a service.

## Test a model

Step 1. Select **AI Studio → Trained Models** from the left navigation pane.

Step 2. Click the name of the model you want to test.

Step 3. Fill in the required information.

- **Test Image**: Click **Browse** and select the image you want to test.
- **Queue**: partition in Slurm
- **CPU Cores**: number of CPU cores
- **GPU**: select whether to use GPU

Step 4. Click **Test**.

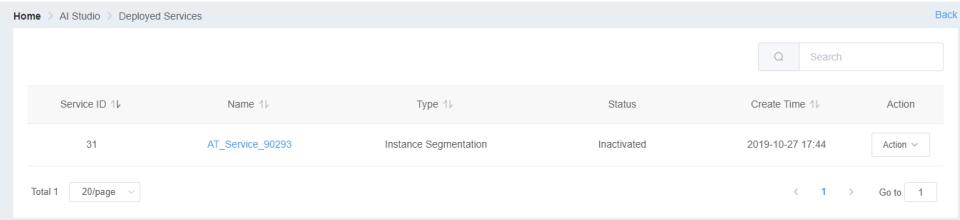
# Deployed services

This section describes how to publish a trained model or deploy it as a service.

## View services

Select **AI Studio → Deployed Services** from the left navigation pane.

The Deployed Services page is displayed.




This page shows details of all the deployed services, and provides some convenient functions such as sorting services by field and searching for services by keyword. It also supports multiple actions on the deployed services, including **Activate**, **Inactivate**, **Delete** and **Detail**. Different states of deployed services are described as follows:

- **Inactivated**: The service is inactivated.
- **Activated**: The service is activated. An activated service is running as a job with the same name as the service name. To ensure that the service is active, the job cannot be stopped.
- **Deploying**: The service is being deployed.

## Inactivate an activated service

Step 1. Select **AI Studio → Deployed Services** from the left navigation pane.

Step 2. Find the activated service you want to inactivate, and then select **Action → Inactivate**.

## Activate an inactivated service

Step 1. Select **AI Studio → Deployed Services** from the left navigation pane.

Step 2. Find the inactivated service you want to activate, and then select **Action → Activate**.
The service turns to the **Deploying** state, during which the service cannot be deleted.

Step 3. Optional: After the service is activated, click **Action → Detail** in the same row as the service on the Deployed Services page.
Details of the service deployment are displayed.

**Note:** Activated services take up a lot of resources. If you do not need a service, inactivate it.

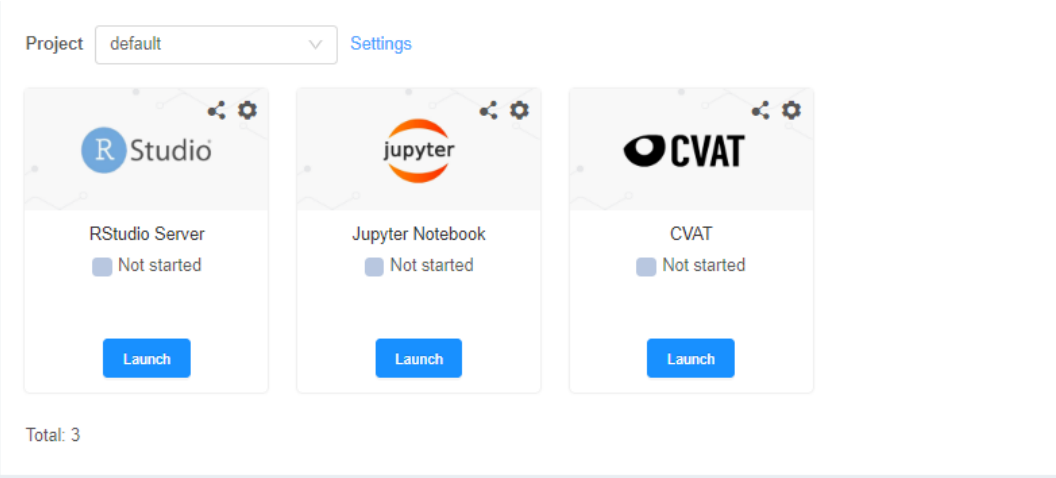
## Use an activated service

Step 1. Select **AI Studio → Deployed Services** from the left navigation pane.

Step 2. Click the name of the activated service you want to use.
The service description is displayed, from which you can learn how to use the service.

# Chapter 5. Cloud Tools

LiCO provides CVAT, Jupyter Notebook, and RStudio Server. You can use this function on the **Cloud Tools** tab.




Select **Cloud Tools** from the left navigation pane.

The Cloud Tools page is displayed.

On this page, you can:

- Create, browse, edit, and delete specific projects.
- Start, edit, share, or stop specific CVAT, Jupyter Notebook, and RStudio Server instances for each project.
- Create default projects.

## Manage projects

### Create a new project

Step 1. Select **Create New Project** from the **Project** drop-down list.

Step 2. Fill in the required information.

- **Name** (required): name of the project
- **Workspace** (required): working directory for the instance
- **Environment** (required): used to store the instance runtime environment. The installed packages in the instance are saved in this directory.

  **Note:** If the workspace is selected for the project, a new environment will be initialized in this directory. You can also initialize a new environment.

Step 3. Click **OK**.

## View a project

Select the target project from the **Project** drop-down list. The corresponding RStudio Server, Jupyter Notebook, and CVAT instances will be displayed.

## Edit a project

Only the project in **Not started** status can be edited.

Step 1. Select the target project from the **Project** drop-down list and click **Settings** on the right. The Project Settings page will be displayed.

Step 2. Fill in the required information.

- **Name** (required): name of the project

Step 3. Click **OK**.

## Delete a project

Only the project in **Not started** status can be deleted.

Step 1. Select the target project from the **Project** drop-down list and click **Settings** on the right. The Project Settings page will be displayed.

Step 2. Click **Delete**. A window will be displayed.

**Attention:** You can select **Delete Environments** to delete the environment.

Step 3. Click **OK**. .

---

# CVAT

Computer Vision Annotation Tool (CVAT) is a free, online, interactive video and image annotation tool for computer vision.

## Start a CVAT instance

Step 1. Find the target CVAT instance, and then select **Launch**.

Step 2. Fill in the required information.

- **Queue** (required): queue selected to run the instance
- **CPU Cores** (required): number of CPU cores to run the instance
- **Login Name** (required): user name for logging in to the instance
- **Login Password** (required): password for logging in to the instance
- **Memory Used (MB)**: the maximum memory required to run the instance.

Step 3. Click **Launch**.

Step 4. Click **Open**.

## Edit a CVAT instance

Step 1. Find the target CVAT instance and click  . The CVAT Settings page will be displayed.

Step 2. Fill in the required information.

- **Queue** (required): queue selected to run the instance

- **CPU Cores** (required): number of CPU cores to run the instance
- **Login Name** (required): user name for logging in to the instance
- **Login Password** (required): password for logging in to the instance
- **Memory Used (MB)**: the maximum memory required to run the instance.

Step 3. Click **Apply**.

Step 4. Restart CVAT instance to make the changes take effect.

## Stop a CVAT instance

Only the CVAT instances in the **Running** or **Queuing** state can be stopped.

Step 1. Find the target CVAT instance and click  .
A dialog box is displayed for you to confirm whether to stop the CVAT instance.

Step 2. Click **YES**.
The current CVAT instance stops running.

## Share a CVAT instance

Only the opened CVAT instances in the **Running** state can be shared.

Step 1. Find the target CVAT instance and click  .
A dialog box is displayed for you to copy on-line platform URL of CVAT instance.

Step 2. Click **Copy To Clipboard**.

Step 3. Open the URL in browser, input user name and password to go to on-line platform of CVAT instance.

---

# Jupyter Notebook

Jupyter Notebook is a web-based interactive computing platform for software development, documentation development, code running, and result sharing.

## Start a Jupyter Notebook instance

Step 1. Find the target Jupyter Notebook instance and click **Launch**.

Step 2. Fill in the required information.

- **Container Image** (required): Select a Jupyter Notebook image from the drop-down list, which needs to be created and uploaded by the user.
- **Jupyter CMD** (required): Input a custom command to start the image. System image is not supported.
- **Login Password** (required): password for logging in to the online Jupyter Notebook editing environment.
- **Queue** (required): queue selected to run the instance.
- **CPU Cores**: number of CPU cores to run the instance.
- **GPU**: number of GPUs to run the instance.
- **GPU Resource Type**: GPU resource type to run the instance.
- **Auto Stop**: the auto stop time of jobs. Select to stop jobs based on the configuration of `/etc/lico/lico.ini.d/cloudtools.ini`.

- **Wall Time**: the maximum execution time of jobs. Enter day, hour, and month in the format “Xd, Xh, Xm”. The default value is 24h.

Step 3. Click **Launch**.

Step 4. Click **Open**.

**Note:** For the first time to create Jupyter Notebook instance, input LiCO user name and Jupyter Notebook password, and click **Sign in**.

## Edit a Jupyter Notebook instance

Step 1. Find the target Jupyter Notebook instance and click  . The Jupyter Notebook Settings page will be displayed.

Step 2. Fill in the required information.

- **Container Image** (required): Select a Jupyter Notebook image from the drop-down list, which needs to be created and uploaded by the user.
- **Jupyter CMD** (required): Input a custom command to start the image. System image is not supported.
- **Login Password** (required): password for logging in to the online Jupyter Notebook editing environment.
- **Queue** (required): queue selected to run the instance.
- **CPU Cores**: number of CPU cores to run the instance.
- **GPU**: number of GPUs to run the instance.
- **GPU Resource Type**: GPU resource type to run the instance.
- **Auto Stop**: the auto stop time of jobs. Select to stop jobs based on the configuration of `/etc/lico/lico.ini.d/cloudtools.ini`.
- **Wall Time**: the maximum execution time of jobs. Enter day, hour, and month in the format “Xd, Xh, Xm”. The default value is 24h.

Step 3. Click **Apply**.

Step 4. Restart Jupyter Notebook instance to make the changes take effect.

## Stop a Jupyter Notebook instance

Only the Jupyter Notebook instances in the **Running** or **Queuing** state can be stopped.

Step 1. Find the target Jupyter Notebook instance and click  .
A dialog box is displayed for you to confirm whether to stop the Jupyter Notebook instance.

Step 2. Click **YES**.
The current Jupyter Notebook instance stops running.

## Share a Jupyter Notebook instance

Only the opened Jupyter Notebook instances in the **Running** state can be shared.

Step 1. Find the target Jupyter Notebook instance and click  .
A dialog box is displayed for you to copy on-line platform URL of Jupyter Notebook instance.

Step 2. Click **Copy To Clipboard**.

Step 3. Open the URL in browser, input user name and password to go to on-line platform of Jupyter Notebook instance.

## RStudio Server

RStudio Server enables you to provide a browser based interface to a version of R running on a remote Linux server, bringing the power and productivity of the RStudio IDE to server-based deployments of R. RStudio Server does not support Chinese version.

## Start a RStudio Server instance

Step 1. Find the target RStudio Server instance, and then select **Launch**.

Step 2. Fill in the required information.

- **Queue** (required): queue selected to run the instance
- **CPU Cores** (required): number of CPU cores to run the instance
- **GPU**: number of GPUs to run the instance
- **GPU Resource Type**: GPU resource type to run the instance
- **Login Password** (required): password for logging in to the online RStudio Server editing environment.
- **Wall Time**: the maximum execution time of jobs. Enter day, hour, and month in the format “Xd, Xh, Xm”. The default value is 24h.

Step 3. Click **Launch**.

Step 4. Click **Open**.

**Note:** For the first time to create RStudio Server instance, input LiCO user name and RStudio Server password, and click **Sign in**.

## Edit a RStudio Server instance

Step 1. Find the target RStudio Server instance and click  . The RStudio Server Settings page will be displayed.

Step 2. Fill in the required information.

- **Queue** (required): queue selected to run the instance
- **CPU Cores** (required): number of CPU cores to run the instance
- **GPU**: number of GPUs to run the instance
- **GPU Resource Type**: GPU resource type to run the instance
- **Login Password** (required): password for logging in to the online RStudio Server editing environment.
- **Wall Time**: the maximum execution time of jobs. Enter day, hour, and month in the format “Xd, Xh, Xm”. The default value is 24h.

Step 3. Click **Apply**.

Step 4. Restart RStudio Server instance to make the changes take effect.

## Stop a RStudio Server instance

Only the RStudio Server instances in the **Running** or **Queuing** state can be stopped.

Step 1. Find the target RStudio Server instance and click  .
A dialog box is displayed for you to confirm whether to stop the RStudio Server instance.

Step 2. Click **YES**.
The current RStudio Server instance stops running.

## Share a RStudio Server instance

Only the opened RStudio Server instances in the **Running** state can be shared.

Step 1. Find the target RStudio Server instance and click  .
A dialog box is displayed for you to copy on-line platform URL of RStudio Server instance.

Step 2. Click **Copy To Clipboard**.

Step 3. Open the URL in browser, input user name and password to go to online platform of RStudio Server instance.

# Chapter 6. Job submission

LiCO 5.3.0 and later versions support to submit jobs through APIs. To learn how to use the APIs, refer to the *LiCO 6.2.0 OpenAPI Guide*.

## Submit an industry standard AI job

Step 1. Select **Submit Job** from the left navigation pane.

Step 2. Select the **Industry Standard AI** tab.

Templates for industry-standard AI jobs are displayed for your selection. All job submission tasks described in this section are performed on this tab page.

## Submit a TensorFlow Single Node job

Step 1. Click **Use** in the TensorFlow Single Node area.

Step 2. Fill in the required information.

- **Job Name** (required): name of the job. For LiCO 6.1.0 and later versions, the initial job name will be automatically created in the format of "template name + time".
- **Workspace** (required): working directory for the job.
- **Container image** (required): Singularity container image used by the job.
- **Program (.py or .sh)** (required): TensorFlow program, which will eventually run the computing task. This program must comply with the TensorFlow standard and be a .py file.
- **Program args**: other parameters required to run the program. Additional parameters should be given in plain text, delineated by spaces.
- **Queue** (required): name of the queue to which the job belongs. Users can only select queues for which they have access permission.
- **Runtime** (required): name of the runtime selected. You can select a runtime you created or use the default value.
- **Exclusive**: indicates whether the job can use all the CPU resources.
- **CPU Cores Per Node** (required): number of CPU cores per node. This parameter is displayed only when the **Exclusive** check box is not selected.
- **GPU Per Node**: number of GPUs per node.
- **GPU Resource Type**: Click the drop-down list to select the target GPU resource type. Both physial GPU and MIG device can be scheduled. The options can only be displayed when the value of **GPU Per Node** is not less than **1** and more than one GPU type can be selected in the cluster.
- **Wall Time**: maximum execution time of the job. It will be stopped after running for more than the configured **Wall Time**.
- **Notify Job Completion**: determines whether to send a notification when the job is completed and the notification method.

Step 3. Click **Submit**.

LiCO has built-in TensorBoard features for TensorFlow programs, helping users monitor the running process of TensorFlow programs on the graphical interface. After a job has been submitted, you can go to the job details page and select the TensorBoard tab. After clicking **Browse** and selecting the export directory for the TensorFlow program, you can do the following:

- Click **View** to launch TensorBoard.
- Click **Refresh** to refresh TensorBoard.
- Click **New window** to display a complete view.

## Submit a TensorFlow Multinode job

Step 1. Click **Use** in the TensorFlow Multinode area.

Step 2. Fill in the required information.

**Nodes** (required): number of nodes used by the job

For the description of other parameters, see "Submit a TensorFlow Single Node job" on page 53.

Step 3. Click **Submit**.

LiCO has built-in TensorBoard features for TensorFlow programs, helping users monitor the running process of TensorFlow programs on the graphical interface. After a job has been submitted, you can go to the job details page and select the TensorBoard tab. After clicking **Browse** and selecting the export directory for the TensorFlow program, you can do the following:

- Click **View** to launch TensorBoard.
- Click **Refresh** to refresh TensorBoard.
- Click **New window** to display a complete view.

## Submit a TensorFlow2 Single Node job

The TensorFlow2 Single Node job template supports program running on one node with CPU or GPU. For distributed training, this template supports MirroredStrategy.

Step 1. Click **Use** in the TensorFlow2 Single Node area.

Step 2. Fill in the required information.

**Program (.py or .sh)** (required): TensorFlow2 program, which will eventually run the computing task. This program must comply with the TensorFlow2 standard.

For the description of other parameters, see "Submit a TensorFlow Single Node job" on page 53.

Step 3. Click **Submit**.

LiCO has built-in TensorBoard features for TensorFlow programs, helping users monitor the running process of TensorFlow2 programs on the graphical interface. After a job has been submitted, you can go to the job details page and select the TensorBoard tab. After clicking **Browse** and selecting the `train_dir` directory for this job, you can do the following:

- Click **View** to launch TensorBoard.
- Click **Refresh** to refresh TensorBoard.
- Click **New window** to display a complete view.

## Submit a TensorFlow2 Multinode job

The TensorFlow2 Multinode job template supports program running on more than one node with CPU or GPU. Using different PS/Worker settings, this template can support MultiWorkerMirroredStrategy and ParameterServerStrategy.

Step 1. Click **Use** in the TensorFlow2 Multinode area.

Step 2. Fill in the required information.

- **Program (.py or .sh)** (required): TensorFlow2 program, which will eventually run the computing task. This program must comply with the TensorFlow2 standard.
- **Nodes** (required): number of nodes used by the job.

For the description of other parameters, see "Submit a TensorFlow Single Node job" on page 53.

**Note:** By adjusting the **PS/Worker Setting** under **Advanced Parameters**, different TensorFlow2 distributed strategies can be used. Default PS/Worker setting indicates MultiWorkerMirroredStrategy, in which mode no PS process is available. To use ParameterServerStrategy, click **Auto** behind **PS/Worker Setting** and then select the **Manual** mode. Under this mode, you can set **PS Count** to determine the number of PS processes, or set **GPU Per Worker** to determine the number of Worker processes.

Step 3. Click **Submit**.

LiCO has built-in TensorBoard features for TensorFlow programs, helping users monitor the running process of TensorFlow2 programs on the graphical interface. After a job has been submitted, you can go to the job details page and select the TensorBoard tab. After clicking **Browse** and selecting the `train_dir` directory for this job, you can do the following:

- Click **View** to launch TensorBoard.
- Click **Refresh** to refresh TensorBoard.
- Click **New window** to display a complete view.

## Submit a Caffe job

Step 1. Click **Use** in the Caffe area.

Step 2. Fill in the required information.

**Program (.py or .sh)** (required): Select a Caffe program file to run.

For the description of other parameters, see "Submit a TensorFlow Single Node job" on page 53.

Step 3. Click **Submit**.

## Submit an Intel Caffe job

Step 1. Click **Use** in the Intel Caffe area.

Step 2. Fill in the required information.

- **Program (.py or .sh)** (required): Select an Intel Caffe program file to run.
- **Nodes** (required): number of nodes used by the job.

For the description of other parameters, see "Submit a TensorFlow Multinode job" on page 54.

Step 3. Click **Submit**.

## Submit a MXNet Single Node job

Step 1. Click **Use** in the MXNet Single Node area.

Step 2. Fill in the required information.

**Program (.py or .sh)** (required): Select a MAXNet program file to run.

For the description of other parameters, see “Submit a TensorFlow Single Node job” on page 53.

Step 3. Click **Submit**.

## Submit a MXNet MultiNode job

Step 1. Click **Use** in the MXNet Multinode area.

Step 2. Fill in the required information.

- **Program (.py or .sh)** (required): Select a MAXNet program file to run.
- **Nodes** (required): number of nodes used by the job.

For the description of other parameters, see “Submit a TensorFlow Single Node job” on page 53.

Step 3. Click **Submit**.

## Submit a Neon job

Step 1. Click **Use** in the Neon area.

Step 2. Fill in the required information.

Step 3. Click **Submit**.

## Submit a Chainer Single Node Job

Step 1. Click **Use** in the Chainer Single Node area.

Step 2. Fill in the required information.

**Program (.py)** (required): Select a Chainer program file to run.

For the description of other parameters, see “Submit a TensorFlow Single Node job” on page 53.

Step 3. Click **Submit**.

## Submit a Chainer Multinode job

Step 1. Click **Use** in the Chainer Multinode area.

Step 2. Fill in the required information.

- **Chainer program (.py)** (required): Select a Chainer program file to run.
- **Nodes** (required): number of nodes used by the job.

For the description of other parameters, see “Submit a TensorFlow Single Node job” on page 53.

Step 3. Click **Submit**.

**Note:** Chainer uses OpenMPI to run multinode jobs. If GPUs are used, each job will run on one GPU; if no GPUs are used, each job will run on one node.

## Submit a PyTorch Single Node job

Step 1. Click **Use** in the Pytorch Single Node area.

Step 2. Fill in the required information.

**Program (.py or .sh)** (required): Select a PyTorch program file to run.

For the description of other parameters, see “Submit a TensorFlow Single Node job” on page 53.

Step 3. Click **Submit**.

## Submit a scikit-learn job

Step 1. Click **Use** in the scikit-learn area.

Step 2. Fill in the required information.

**Program (.py or .sh)** (required): Select a scikit-learn program file to run.

For the description of other parameters, see "Submit a TensorFlow Single Node job" on page 53.

Step 3. Click **Submit**.

## Submit a Nvidia TensoRT job

Step 1. Click **Use** in the **Nvidia TensoRT** area.

Step 2. Fill in the required information

- **Program (.py or .sh)** (required): Select the Nvidia TensorRT program file to be executed.

**Note:** For the description of other parameters, see "Submit a TensorFlow Single Node job" on page 53.

Step 3. Click **Submit**.

---

## Submit an HPC job

Step 1. Select **Submit Job** from the left navigation pane.

Step 2. Select the **HPC** tab.

Templates for HPC jobs are displayed for your selection. All job submission tasks described in this section are performed on this tab page.

## Submit an MPI job

An MPI job is a distributed computing job using the MPI standard.

Step 1. Click **Use** in the MPI area.

Step 2. Fill in the required information.

- **Job Name** (required): job name. For LiCO 6.1.0 and later versions, the initial job name will be automatically created in the format of "template name + time".
- **Workspace** (required): working directory of the job. Job output files will be saved in this directory.
- **Runtime** (required): name of the runtime selected. You can select a runtime you created or use the default value.
- **MPI program** (required): the MPI program, that is, program that will eventually run the computing task. For the program to carry out parallel computing on multiple servers, it must comply with the MPI standard.
- **MPI Environment**: MPI environment variables.
- **MPI Run Arguments**: parameters for running mpirun. Additional parameters should be given in plain text, delineated by spaces.
- **Queue** (required): the name of the queue to which the job belongs. Users can only select queues for which they have access permission.

- **Nodes** (required): number of computing nodes required to run the program.
- **CPU Cores Per Node** (required): the number of CPU cores required by each computing node to run the program.
- **GPU Per Node**: number of GPUs per node for training.
- **Memory Used**: the memory required to run the program.
- **Wall Time**: maximum execution time of the job. It will be stopped after running for more than the configured **Wall Time**.
- **EAR**: required if EAR is enabled, Provide users with the options of setting EAR parameters when EAR is enabled. The options include:
  - **Auto**: The default policy without EAR-related options. After the job is submitted, the EAR will be used according to the situation configured by the administrator.
  - **Energy Policy**: If this option is displayed, EAR is enabled. The options include:
    - **Default**: Use the default rules configured by the system.
    - **Min Time**: Use the minimum time rule of the system. There is large energy consumption under this rule.
    - **Min Energy**: Use the minimum energy consumption strategy.
  - **Energy Tag**: The cursor label accessible in all configurations.
  - **OFF**: EAR is not used explicitly, and the MPI program is started through mpirun or other mpi built-in programs. When EAR is configured through slurm, the option for closing EAR is added to the job file.
  - **EAR Settings**:
    - **MPI Type**: Users should specify the MPI type, and the default value is **Default**. If **Default** is selected, the MPIDEFAULT value will be configured for Slurm.
    - **Log**: Whether to display EAR log.

Step 3. Click **Submit**.

## Submit an ANSYS job

An ANSYS job is a job using the ANSYS software, where users can run engineering simulations.

Step 1. Click **Use** in the ANSYS area.

Step 2. Fill in the required information.

- **GUI mode**: Click **GUI mode** to run the ANSYS job in GUI mode. Once the job is submitted, a VNC session will be launched and the ANSYS graphical interface will begin to run in the VNC session. It can also be accessed via the VNC management.
- **ANSYS Binary** (required): the location of the ANSYS software.
- **ANSYS Environment**: the environment variables required when running ANSYS.
- **ANSYS Run Arguments**: parameters for running ANSYS. Additional parameters should be given in plain text, delineated by spaces.
- **ANSYS Input**: the file imported when running the program.
- **ANSYS Output**: the file exported when running the program.

**Note:** For the description of other parameters, see “Submit an MPI job” on page 57.

Step 3. Click **Submit**.

## Submit a COMSOL job

A COMSOL job is a job using the COMSOL software, where users can model and simulate physics-based problems.

Step 1. Click **Use** in the COMSOL area.

Step 2. Fill in the required information.

- **GUI Mode**: Click **GUI Mode** to run the COMSOL job in GUI mode. Once the job has been submitted, a VNC session will be launched and the COMSOL graphical interface will begin to run in the VNC session. It can also be accessed via the VNC management.
- **COMSOL Binary** (required): the location of the COMSOL software.
- **COMSOL Environment**: the environment variables required when running COMSOL.
- **COMSOL Run Arguments**: parameters for running COMSOL. Additional parameters should be given in plain text, delineated by spaces.
- **COMSOL Input**: the file imported when running the program.
- **COMSOL Output**: the file exported when running the program.

**Note:** For the description of other parameters, see "Submit an MPI job" on page 57.

Step 3. Click **Submit**.

## Submit a Charliecloud MPI job

Charliecloud provides user-defined software stacks (UDSS) for high-performance computing (HPC) centers. Container images can be built by using Docker or any else tools that can generate a standard Linux filesystem tree.

Step 1. Click **Use** in the Charliecloud MPI area.

Step 2. Fill in the required information.

**Container Image Path** (required): path of the Charliecloud container image.

For the description of other parameters, see "Submit an MPI job" on page 57.

Step 3. Click **Submit**.

## Submit a Singularity MPI job

Singularity is a container platform. It allows you to create and run containers that package up pieces of software in a way that is portable and reproducible. You can build a container using Singularity on your laptop, and then run it on many of the largest HPC clusters in the world, on local university or company clusters, on a single server, in the cloud, or on a workstation down the hall.

Step 1. Click **Use** in the Singularity MPI area.

Step 2. Fill in the required information.

**Container Image** (required): Singularity container image.

For the description of other parameters, see "Submit an MPI job" on page 57.

Step 3. Click **Submit**.

## Submit an Intel oneAPI job

Intel oneAPI is an open-source and standard programming model designed for all industries, which provides the uniform service for the developers of CPU, GPU, and FPGA accelerators. Based on industrial standard and existing programming model of developers, oneAPI open standard can be widely used in varied structures and hardware from different suppliers. The use of Intel oneAPI improves the performance of MPI, OpenMP, TensorFLow, Pythorch, and other programs.

- Intel MPI: Intel® MPI Library is a multifabric message-passing library that implements the open-source MPICH specification. Use the library to create, maintain, and test advanced, complex applications that perform better on high-performance computing (HPC) clusters based on Intel® processors. For more information, go to: https://www.intel.cn/content/www/cn/zh/developer/tools/oneapi/mpi-library.html#gs.fvsp6j
- Intel OpenMP: Intel OpenMP enables users to run Intel oneAPI DPC++/C++ through Intel OpenMP* library, improving OpenMP program performance. For more information, go to: https://www.intel.com/content/www/us/en/develop/documentation/oneapi-dpcpp-cpp-compiler-dev-guide-and-reference/top/optimization-and-programming-guide/openmp-support/openmp-library-support/using-the-openmp-libraries.html
- Intel VTune: Intel VTune Profiler, provided with Intel snapshot performance analyzer, enables users to analyze the serial and multi-threaded applications in hardware platforms (CPU, GPU, FPGA), and analyze the local and remote targets in Windows*, Linux*, and Android*. For more information, go to: https://www.intel.com/content/www/us/en/develop/documentation/vtune-help/top.html
- Intel Modin：The Intel® Distribution of Modin* is a performant, parallel, and distributed dataframe system that is designed around enabling data scientists to be more productive with the tools that they love. This library is fully compatible with the pandas API. It is powered by OmniSci* in the back end and provides accelerated analytics on Intel® platforms. For more information, go to: https://www.intel.com/content/www/us/en/developer/tools/oneapi/distribution-of-modin.html
- Intel Distribution for Python：Intel® Distribution for Python* is included as part of the Intel® oneAPI AI Analytics Toolkit, which provides accelerated machine learning and data analytics pipelines with optimized deep-learning frameworks and high-performing Python libraries. For more information, go to: https://www.intel.com/content/www/us/en/developer/tools/oneapi/distribution-for-python.html?wapkw=intel%20python

**Note:** This function is unavailable if Intel oneAPI is not installed.

Step 1. Select **Submit Job** from the left navigation pane.

Step 2. Select the **Intel oneAPI** tab.

Templates for Intel oneAPI jobs are displayed for your selection. All job submission tasks described in this section are performed on this tab page.

## Submit an Intel VTune Profiler job

Step 1. Click **Use** in the Intel VTune Profiler area.

Step 2. Fill in the required information.

- **Job Name** (required): job name.
- **Workspace** (required): working directory of the job. Job output files will be saved in this directory.
- **Runtime ENV**: name of the runtime selected. You can select a runtime you created or use the default value.
- **Data Directory**: the directory storing the running VTune files and the corresponding reports.

- **Queue** (required): the name of the queue to which the job belongs. Users can only select queues for which they have access permission.
- **CPU Cores** (required): number of CPU cores in each node.

For the description of other parameters, see "Submit an Intel MPI job" on page 63.

Step 3. Click **Submit**.

## Submit an Intel Optimization for PyTorch Single Node job

Step 1. Click **Use** in the Intel Optimization for PyTorch Single Node area.

Step 2. Fill in the required information.

- **Job Name** (required): job name.
- **Workspace** (required): working directory of the job. Job output files will be saved in this directory.
- **Runtime ENV**: name of the runtime selected. You can select a runtime you created or use the default value. The default value is Intel_Optimization_for_PyTorch.
- **Program (.py or .sh)** (required): PyTorch program.
- **Program Args**: the required parameters of PyTorch program.
- **Queue** (required): name of the queue to which the job belongs. Users can only select queues for which they have access permission.
- **Exclusive**: indicates whether the job can use all the CPU resources. If selected, the cores in this node cannot be used by other programs; if not selected, the cores in this node can be used by other programs
- **CPU Cores Per Node** (required): number of CPU cores per node. This parameter is displayed only when the **Exclusive** check box is not selected.
- **Wall Time**: maximum execution time of the job. It will be stopped after running for more than the configured **Wall Time**.
- **Notify Job Completion**: determines whether to send a notification when the job is completed and the notification method.

Step 3. Click **Submit**.

## Submit an Intel Optimization for TensorFlow2 Single Node job

Step 1. Click **Use** in the Intel Optimization for TensorFlow2 Single Node area.

Step 2. Fill in the required information.

- **Job Name** (required): job name.
- **Workspace** (required): working directory of the job. Job output files will be saved in this directory.
- **Runtime ENV**: name of the runtime selected. You can select a runtime you created or use the default value. The default value is `Intel_Optimization_for_TensorFlow`.
- **Program (.py or .sh)** (required): TensorFlow program.
- **Program Args**: the required parameters of TensorFlow program.
- **Queue** (required): name of the queue to which the job belongs. Users can only select queues for which they have access permission.
- **Exclusive**: indicates whether the job can use all the CPU resources. If selected, the cores in this node cannot be used by other programs; if not selected, the cores in this node can be used by other programs

- **CPU Cores Per Node** (required): number of CPU cores per node. This parameter is displayed only when the **Exclusive** check box is not selected.
- **Wall Time**: maximum execution time of the job. It will be stopped after running for more than the configured **Wall Time**.
- **Notify Job Completion**: determines whether to send a notification when the job is completed and the notification method.

Step 3. Click **Submit**.

LiCO has built-in TensorBoard features for TensorFlow programs, helping users monitor the running process of TensorFlow programs on the graphical interface. After a job has been submitted, you can go to the job details page and select the TensorBoard tab. After clicking **Browse** and selecting the export directory for the TensorFlow program, you can do the following:

- Click **View** to launch TensorBoard.
- Click **Refresh** to refresh TensorBoard.
- Click **New window** to display a complete view.

## Submit an Intel Optimization for TensorFlow2 Multi Node job

Step 1. Click **Use** in the Intel Optimization for TensorFlow2 Multi Node area.

Step 2. Fill in the required information.

- **Job Name** (required): job name.
- **Workspace** (required): working directory of the job. Job output files will be saved in this directory.
- **Runtime ENV**: name of the runtime selected. You can select a runtime you created or use the default value. The default value is `Intel_Optimization_for_TensorFlow`.
- **Program (.py or .sh)** (required): TensorFlow2 program.
- **Program Args**: the required parameters of TensorFlow2 program.
- **Queue** (required): name of the queue to which the job belongs. Users can only select queues for which they have access permission.
- **Nodes** (required): number of computing nodes required to run the program.
- **Exclusive**: indicates whether the job can use all the CPU resources. If selected, the cores in this node cannot be used by other programs; if not selected, the cores in this node can be used by other programs
- **CPU Cores Per Node** (required): number of CPU cores per node. This parameter is displayed only when the **Exclusive** check box is not selected.
- **Wall Time**: maximum execution time of the job. It will be stopped after running for more than the configured **Wall Time**.
- **Notify Job Completion**: determines whether to send a notification when the job is completed and the notification method.

  **Note:** By adjusting the **PS/Worker Setting** under **Advanced Parameters**, different TensorFlow2 distributed strategies can be used. Default PS/Worker setting indicates MultiWorkerMirroredStrategy, in which mode no PS process is available. To use ParameterServerStrategy, click **Auto** behind **PS/Worker Setting** and then select the **Manual** mode. Under this mode, you can set **PS Count** to determine the number of PS processes.

Step 3. Click **Submit**.

LiCO has built-in TensorBoard features for TensorFlow programs, helping users monitor the running process of TensorFlow programs on the graphical interface. After a job has been submitted, you can go to the job details page and select the TensorBoard tab. After clicking **Browse** and selecting the export directory for the TensorFlow program, you can do the following:

- Click **View** to launch TensorBoard.
- Click **Refresh** to refresh TensorBoard.
- Click **New window** to display a complete view.

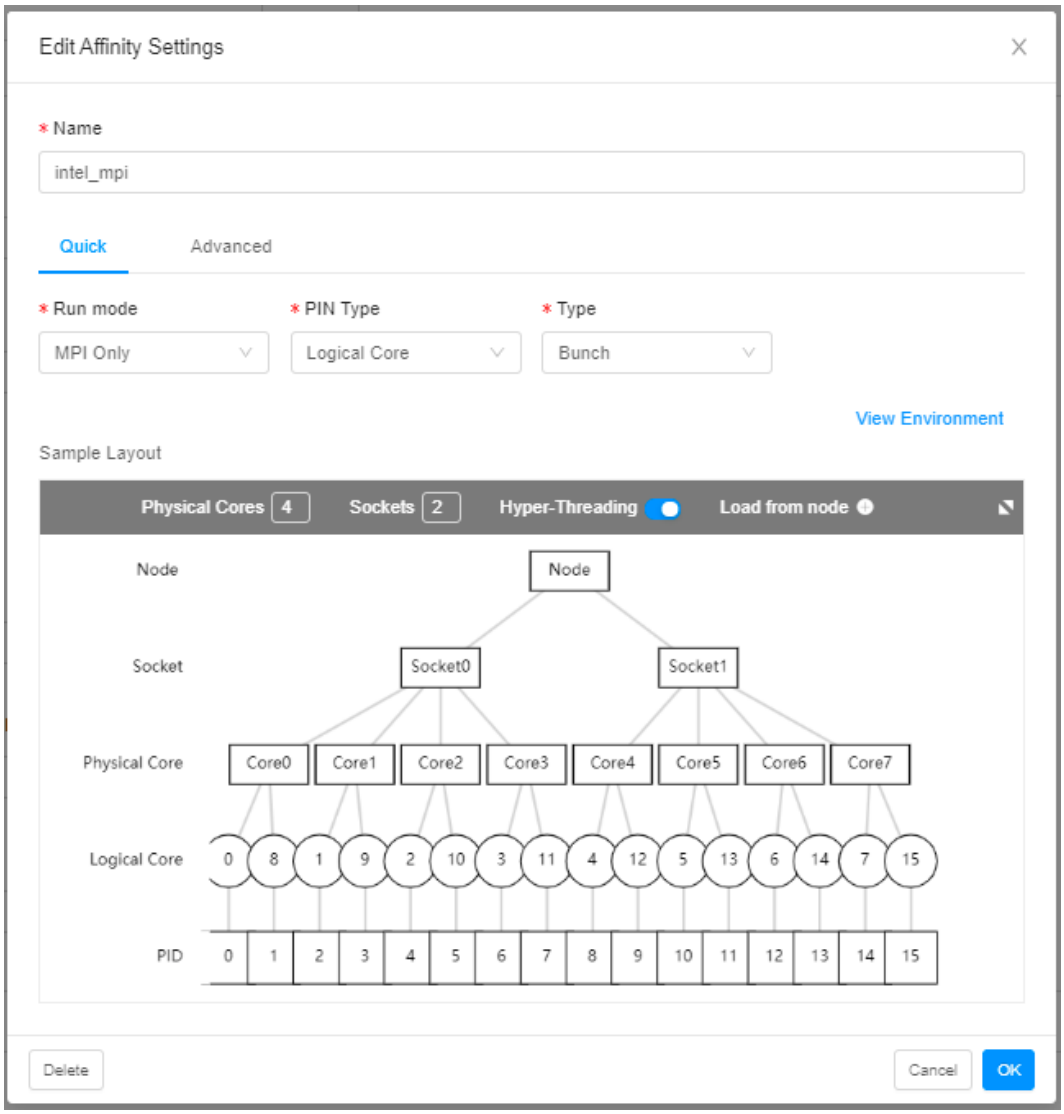
## Submit an Intel MPI job

Step 1. Click **Use** in the Intel MPI area.

Step 2. Fill in the required information.

- **Job Name** (required): job name.
- **Workspace** (required): working directory of the job. Job output files will be saved in this directory.
- **Runtime ENV**: name of the runtime selected. You can select a runtime you created or use the default value.
- **Program** (required): the MPI program, that is, program that will eventually run the computing task. For the program to carry out parallel computing on multiple servers, it must comply with the MPI standard.
- **MPI Environment**: MPI environment variables.
- **MPIRun Arguments**: parameters for running mpirun. Additional parameters should be given in plain text, delineated by spaces.
- **Program Arguments**: parameters passed to the MPI program in the method defined by the program itself.
- **MPITune Result**: MPITune configuration file output upon MPITune job submission.
- **Intel Performance Analyzer**: Analyzer for analyzing the performance of running jobs. You can select the target analyzer from the drop-down list.

  **Note:** To view the Intel Performance Analyzer report, refer to "Intel Performance Analyzer Report" on page 75.
- **VTune Analysis Type**: required if **VTune Profiler** is selected. You can select the type of running job for **VTune Profiler**.
- **Queue** (required): the name of the queue to which the job belongs. Users can only select queues for which they have access permission.
- **Nodes** (required): number of computing nodes required to run the program.
- **CPU Cores Per Node** (required): the number of CPU cores required by each computing node to run the program. This parameter is available in non-exclusive mode.
- **Affinity**: CPU binding relationship. This parameter is available only in exclusive mode. You can select **Not Bind** or **New Setting**, or click **Edit** to edit the binding settings created before. Parameters on the affinity settings page are described as follows:




**Note:** **Sample Layout** stimulates the CPU binding relationship based on your settings and is for reference only. The actual running results prevail.

- **Name** (required): name of the CPU binding settings.
- **Quick** tab: This tab allows you to create CPU binding settings quickly.
  - **Run Mode** (required): CPU binding for the MPI program only or for the MPI and OpenMP programs.
  - **PIN Type** (required): binding method for **MPI Only** mode, which indicates what cores can be used by the program.
  - **OMP Threads** (required): number of threads enabled for each process in **MPI with OpenMP** mode.
  - **Type** (required): type of binding relationship.
  - **View Environment**: You can click this link to view the environment variable created based on your settings.
  - **Physical Cores**: number of physical cores for a socket in the simulated preview.
  - **Sockets**: number of sockets for a node in the simulated preview.

  - **Hyper-Threading**: indicates whether to enable hyper-threading for nodes in the simulated preview.
  - **Load from node**: You can enter a host name to obtain its actual CPU configurations, and then adjust the values of **Physical Cores**, **Sockets**, and **Hyper-Threading** as required.
- **Advanced** tab: This tab allows you to add other environment variables as required.

- **Memory Used**: the memory required to run the program.
- **Wall Time**: maximum execution time of the job. It will be stopped after running for more than the configured **Wall Time**.
- **Notify Job Completion**: determines whether to send a notification when the job is completed and the notification method.

Step 3. Click **Submit**.

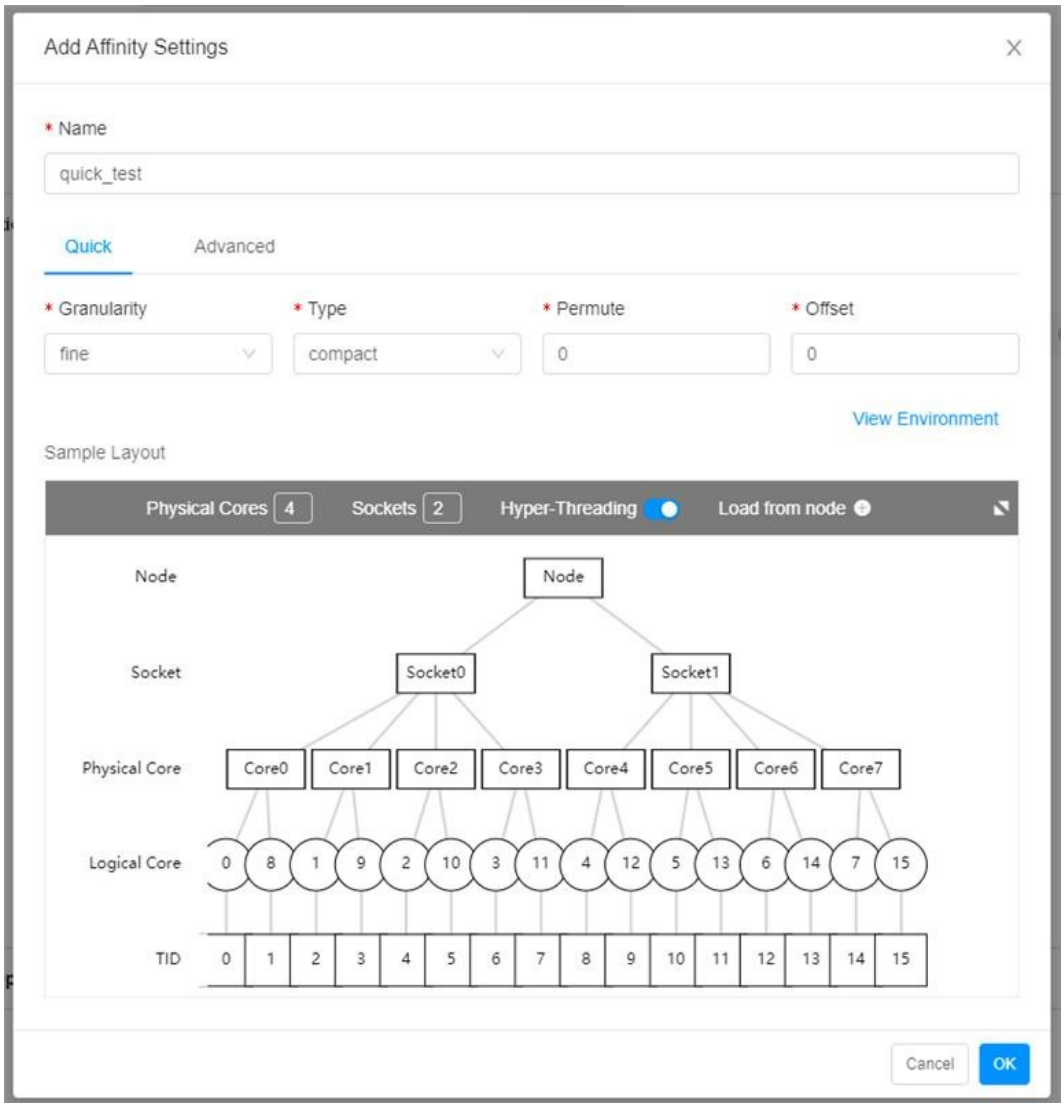
## Submit an Intel OpenMP job

Step 1. Click **Use** in the Intel OpenMP area.

Step 2. Fill in the required information.

- **Runtime ENV** (required): name of the runtime selected. You can select a runtime you created or use the default value.
- **OpenMP Environment**: OpenMP environment variables.
- **OpenMP Program** (required): the OpenMP program, that is, program that will eventually run the computing task.
- **Program Arguments**: parameters for running the OpenMP program.
- **Intel Performance Analyzer**: Analyzer for analyzing the performance of running jobs. You can select the target analyzer from the drop-down list.

  **Note:** To view the Intel Performance Analyzer report, refer to "Intel Performance Analyzer Report" on page 75.
- **VTune Analysis Type**: required if **VTune Profiler** is selected. You can select the type of running job for **VTune Profiler**.
- **Affinity**: CPU binding relationship. This parameter is available only in exclusive mode. You can select **Not Bind** or **New Setting**, or click **Edit** to edit the binding settings created before. Parameters on the affinity settings page are described as follows:




**Note: Sample Layout** stimulates the CPU binding relationship based on your settings and is for reference only. The actual running results prevail.

- **Name** (required): name of the CPU binding settings.
- **Quick** tab: This tab allows you to create CPU binding settings quickly.
  - **Granularity** (required): lowest level of floating allowable for OpenMP threads in the topology. The default value is **fine**.
  - **Type** (required): type of binding relationship between OpenMP threads and CPUs. The default value is **compact**. To use other types, go to the **Advanced** tab.
  - **Permute** (required): permuting level, which is an integer from 0 to 9. The default value is **0**.
  - **Offset** (required): offset value, which is an integer from 0 to 128. The default value is **0**.

For the description of other parameters, see “Submit an Intel MPI job” on page 63.

Step 3. Click **Submit**.

## Submit an Intel MPITune job

Step 1. Click **Use** in the MPI area.

Step 2. Fill in the required information.

- **MPITune Environment**: MPITune environment variables.
- **Config File** (required): configuration file required for running MPITune.

For the description of other parameters, see "Submit an Intel MPI job" on page 63.

Step 3. Click **Submit**.

## Submit an Intel Distribution for Python job

Step 1. Click **Use** in the Intel Distribution for Python area.

Step 2. Fill in the required information.

- **Job Name** (required): job name.
- **Workspace** (required): working directory of the job. Job output files will be saved in this directory.
- **Runtime ENV**: name of the runtime selected. You can select a runtime you created or use the default value `Intel_distribution_for_Python`.
- **Python Program** (required): the Python program to be used. Users should modify Python Pandas code following the Help information on the upper-right corner.
- **Program Arguments**: parameters for running the Python program.
- **Intel Performance Analyzer**: the analyzer for analyzing the performance of running job. Users can select the target analyzer from the drop-down list.

  **Note:** To check the Intel Performance Analyzer report, refer to "Intel Performance Analyzer Report" on page 75.
- **VTune Analysis Type**: required if **VTune Profiler** is selected. You can select the type of running job for **VTune Profiler**.
- **Queue** (required): the name of the queue to which the job belongs. Users can only select queues for which they have access permission.
- **Exclusive**: indicates whether the job can use all the CPU resources. If selected, the cores in this node cannot be used by other programs; if not selected, the cores in this node can be used by other programs.
- **CPU Cores** (required): number of CPU cores in each node.
- **Memory Used(MB)**: the memory required to run the program.
- **Wall Time**: maximum execution time of the job. It will be stopped after running for more than the configured **Wall Time**.
- **Notify Job Completion**: determines whether to send a notification when the job is completed and the notification method.

Step 3. Click **Submit**.

## Submit an Intel Distribution of Modin job

Step 1. Click **Use** in the Intel Distribution of Modin area.

Step 2. Fill in the required information.

- **Job Name** (required): job name.

- **Workspace** (required): working directory of the job. Job output files will be saved in this directory.
- **Runtime ENV**: name of the runtime selected. You can select a runtime you created or use the default value `Intel_Distribution_of_Modin`.
- **Python Program** (required): the Python program to be used. Users should modify Python Pandas code following the Help information on the upper-right corner.
- **Program Arguments**: parameters for running the Python program.
- **Queue** (required): the name of the queue to which the job belongs. Users can only select queues for which they have access permission.
- **Exclusive**: indicates whether the job can use all the CPU resources. If selected, the cores in this node cannot be used by other programs; if not selected, the cores in this node can be used by other programs
- **CPU Cores** (required): number of CPU cores in each node.
- **Memory Used(MB)**: the memory required to run the program.
- **Wall Time**: maximum execution time of the job. It will be stopped after running for more than the configured **Wall Time**.
- **Notify Job Completion**: determines whether to send a notification when the job is completed and the notification method.

Step 3. Click **Submit**.

## Submit an Intel Distribution of Modin Multi Node job

Step 1. Click **Use** in the Intel Distribution of Modin Multi Node area.

Step 2. Fill in the required information.

- **Job Name** (required): job name.
- **Workspace** (required): working directory of the job. Job output files will be saved in this directory.
- **Runtime ENV**: name of the runtime selected. You can select a runtime you created or use the default value `Intel_Distribution_of_Modin`.
- **Python Program** (required): the Python program to be used. Users should modify Python Pandas code following the Help information on the upper-right corner.
- **Program Arguments**: parameters for running the Python program.
- **Queue** (required): the name of the queue to which the job belongs. Users can only select queues for which they have access permission.
- **Nodes** (required): number of computing nodes required to run the program.
- **Exclusive**: indicates whether the job can use all the CPU resources. If selected, the cores in this node cannot be used by other programs; if not selected, the cores in this node can be used by other programs
- **CPU Cores Per Node**: Required if **Exclusive** is not selected. The number of available CPU cores in each train node.
- **Memory Used(MB)**: the memory required to run the program.
- **Wall Time**: maximum execution time of the job. It will be stopped after running for more than the configured **Wall Time**.
- **Notify Job Completion**: determines whether to send a notification when the job is completed and the notification method.

Step 3. Click **Submit**.

## Submit a general job

Step 1. Select **Submit Job** from the left navigation pane.

Step 2. Select the **General** tab.

Templates for general jobs are displayed for your selection. All job submission tasks described in this section are performed on this tab page.

## Submit a general job

A general job is a computing task where the user fully controls the running mode, resource usage, and job file.

Step 1. Click **Use** in the General Job area.

Step 2. Fill in the job name, and click **Browse** to select a job file.

Step 3. Click **Submit**.

## Submit a common job

A common job is a script job that the user can quickly write and run from the Web interface.

Step 1. Click **Use** in the Common Job area.

Step 2. Fill in the job name.

Step 3. Click **Browse** to select a work directory.

Step 4. Type the script to be executed in the script editor (Linux Shell commands are supported).

Step 5. Select the resources needed to run the script.

Step 6. Click **Submit**.

## Submit a Charliecloud job

Charliecloud provides UDSS for HPC centers. Container images can be built by using Docker or any else tools that can generate a standard Linux filesystem tree.

Step 1. Click **Use** in the Charliecloud area.

Step 2. Fill in the job name.

Step 3. Click **Browse** to select a work directory.

Step 4. Click **Browse** to select the path of the Charliecloud container image.

Step 5. Type the script to be executed in the script editor (Linux Shell commands are supported).

Step 6. Select the resources needed to run the script.

Step 7. Click **Submit**.

## Submit a Singularity job

Singularity is a container platform. It allows you to create and run containers that package up pieces of software in a way that is portable and reproducible. You can build a container using Singularity on your laptop, and then run it on many of the largest HPC clusters in the world, on local university or company clusters, on a single server, in the cloud, or on a workstation down the hall.

Step 1. Click **Use** in the Singularity area.

Step 2. Fill in the job name.

Step 3. Click **Browse** to select a work directory.

Step 4. Click **Browse** to select the path of the Singularity container image.

Step 5. Type the script to be executed in the script editor (Linux Shell commands are supported).

Step 6. Select the resources needed to run the script.

Step 7. Click **Submit**.

## Submit a VNC Desktop job

Step 1. Click **Use** in the VNC Desktop area.

Step 2. Fill in the required information.

- **Job Name** (required): job name.
- **Workspace** (required): working directory of the job. Job output files will be saved in this directory.
- **Runtime ENV**: name of the runtime selected. You can select a runtime you created or use the default value.
- **Password** (required): password for logging in to vnc.
- **Queue** (required): the name of the queue to which the job belongs. Users can only select queues for which they have access permission.
- **CPU Cores Per Node** (required): number of CPU cores per node. This parameter is displayed only when the **Exclusive** check box is not selected.
- **GPU Per Node**: number of GPUs per node. The default value is 0.
- **Wall Time**: maximum execution time of the job. It will be stopped after running for more than the configured **Wall Time**.
- **Notify Job Completion**: determines whether to send a notification when the job is completed and the notification method.

Step 3. Click **Submit**.

# Chapter 7. Manage the job lifecycle

**Note:** The status displayed on the page is “Completed” as long as slrum is submitted to a job successfully, no matter the result of the job is “Success” or “Failure”.

## Cancel a job

Step 1. Select **Jobs** from the left navigation pane.

Step 2. Click **Running** or **Waiting** on the status bar.
All running or waiting jobs are displayed in a list.

Step 3. Find the job you want to cancel, and select **Action → Cancel** .

Step 4. Click **Yes**.

The job is canceled.

Canceled jobs are available in the Completed list.

## Re-run a job

Step 1. Select **Jobs** from the left navigation pane.

Step 2. Click **Completed** on the status bar.
All completed jobs are displayed in a list.

Step 3. Find the job you want to re-run, and select **Action → Rerun**.

Step 4. Click **Yes**.

The job is re-executed.

## Copy a job

Step 1. Select **Jobs** from the left navigation pane.

Step 2. Click **Completed** on the status bar.
All completed jobs are displayed in a list.

Step 3. Find the job you want to copy, and select **Action → Copy**.
A page for creating a job is displayed. All the parameters have been already configured in the same way as the selected job. Follow the steps to finish the submission.

## Delete a job

Step 1. Select **Jobs** from the left navigation bar.

Step 2. Click **Completed** on the status pane.
All completed jobs are displayed in a list.

Step 3. Find the job you want to delete, and select **Action → Delete**.
A dialog is displayed, asking you to confirm the action.

Step 4. Click **Yes** .

The job is deleted.

## Job tag

Job tags enable users to identify different jobs, or to filter jobs by tag. A maximum of 10 tags can be added to a job, with one tag containing no more than 20 characters.

## Add tags to a job

Step 1. Select **Jobs** from the left navigation bar.

Step 2. Click the name of a job to enter its details page.

Step 3. On the job details page, click the **Information** tab.

Step 4. Click **Add Tags**.
The Add Tags dialog is displayed.

Step 5. Enter or select one or more tags as required.

**Note:** A maximum of 10 tags are supported. A tag must contain no more than 20 characters, with both Chinese and English characters supported.

Step 6. Click **OK** .

## Clear tags for a job

Step 1. Select **Jobs** from the left navigation bar.

Step 2. Click the name of a job to enter its details page.

Step 3. On the job details page, click the **Information** tab.

Step 4. Click **Clean Tags**.

**Note:** **Clean Tags** will not be displayed if the selected job has no tags.

Step 5. Click **OK** .

## Add the same tags to multiple jobs

Step 1. Select **Jobs** from the left navigation bar.

Step 2. From the job list, select multiple jobs to add the same tags.

Step 3. Click **Add Tags**.
The Add Tags dialog is displayed.

Step 4. Enter or select one or more tags as required.

**Note:** A maximum of 10 tags are supported. A tag must contain no more than 20 characters, with both Chinese and English characters supported.

Step 5. Click **OK** .

## Clear tags for multiple jobs

Step 1. Select **Jobs** from the left navigation bar.

Step 2. From the job list, select multiple jobs to clear tags.

Step 3. Click  behind **Add Tags** and then select **Clear Tags**.

Step 4. Click **OK** .

## Filter jobs by tag

Step 1. Select **Jobs** from the left navigation bar.

Step 2. Select or enter tags in the **Filter Tags** field.

> **Note:** If multiple tags are specified as the filter condition, a job will be listed as long as it contains any of the tags specified. One tag is for exact match, not for fuzzy match.
>
> Jobs with any of the tags specified will be listed.

---

# Job comment

Job comments are used to record job information. From the job list, users can identify jobs and view job information through job comments, such as job parameters specified, job running results, and the purpose for running a job.

## View comments of a job

You can view comments of a job from the job list or on its job details page.

### From the job list

Step 1. Select **Jobs** from the left navigation bar.

Step 2. From the job list, hover your mouse cursor over a job to view its name and comments.

> **Note:** If the job has no comments, only its name will be displayed.

### On the job details page

Step 1. Select **Jobs** from the left navigation bar.

Step 2. Click the name of a job to enter its details page.

Step 3. On the job details page, click the **Information** tab.

Job comments will be displayed on this tab.

## Edit comments of a job

You can edit comments of a job from the job list or on its job details page.

### From the job list

Step 1. Select **Jobs** from the left navigation bar.

Step 2. Find the job for which you want to edit its comments, and select **Action → Comment**.
A page for editing the job comments is displayed.

Step 3. Edit the job comments as required.

Step 4. Click **OK**.

### On the job details page

Step 1. Select **Jobs** from the left navigation bar.

Step 2. Click the name of a job to enter its details page.

Step 3. On the job details page, click the **Information** tab.

Step 4. Click **Edit**.
A page for editing the job comments is displayed.

Step 5. Edit the job comments as required.

Step 6. Click **OK**.

## Process monitoring

You can view process information of a running job on its job details page.

Step 1. Select **Jobs** from the left navigation bar.

Step 2. Click the name of a job to enter its details page.

Step 3. On the job details page, click the **Process** tab.
The process details page is displayed.

On this process details page, you can view the CPU utilization, memory usage, GPU utilization (if multiple GPU nodes are used, the utilization of each GPU node will be displayed if you hover the mouse cursor over the GPU percentage), run time, and run program for a process.

- Drop-down list: Select a specific node to view its process information.
- Auto refresh: After you select a node, information on the process details page will be automatically refreshed every 30s.
- Search box: Search by any field is supported.

## EAR User Report

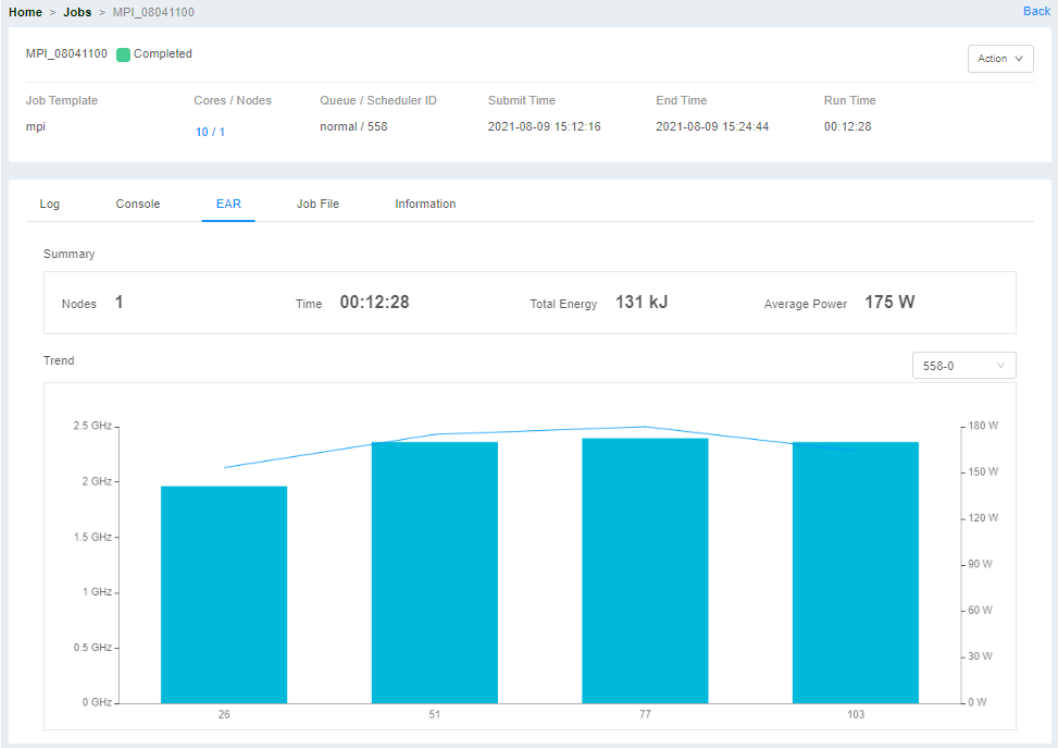
Step 1. Select **Jobs** from the left navigation pane.

Step 2. Click **Completed** on the status bar.

Step 3. Find the job using the EAR policy and click the name of the job. The job page is displayed.

Step 4. On the job page, click the **EAR** tab. You can view the following information:

- In the **Summary** area:
  - **Node**: The number of nodes according to EAR statistics.
  - **Time**: The energy consumption duration according to EAR statistics.
  - **Total Energy**: The total energy consumption according to EAR statistics.
  - **Average Power**: The average power according to EAR statistics.
- In the **Trend** area:
  - X Axis: the number of iterations of the program.
  - Y Axis(left): the average frequency, illustrated by the line chart.
  - Y Axis(right): the average power, illustrated by the bar chart.




## Intel Performance Analyzer Report

Step 1. Select **Jobs** from the left navigation pane.

Step 2. Click **Completed** on the status bar.

Step 3. Find the completed job and click the name of the job. The job page is displayed.

Step 4. On the job page, click **Intel Performance Analyzer**. Do one or more of the following:

- Click **View** to view the simple report containing analysis results.
- Click **Download** to download the simple report containing analysis results.
- Click **Go To VTune** to view the details of analysis results.

  **Note:** This option is only available when **VTune Profiler** is selected.
- Click **Go To VNC** to view the details of analysis results through running commands in the VNC page.

  **Note:** This option is only available when **Trace Analyzer and Collector** or **Advisor** is selected.
- Click **Clear Analysis Data** to clear the data collected by analyzer.

  **Attention:** This step also clears all analysis results.

## Resource monitoring

You can view resource information of a running job on its job details page.

Step 1. Select **Jobs** from the left navigation bar.

Step 2. Click the name of a job to enter its details page.

Step 3. On the job details page, click the **Resource** tab.

The resource details page is displayed.

On this resource details page, you can view the real-time resource occupation data of all nodes running the job, including utilization of CPU, CPU memory, GPU, and GPU memory.

## CPU resource monitoring

Step 1. On the resource details page, select the **CPU** tab. Then select whether to check CPU or memory utilization of the job in the upper right corner.

Step 2. Place the mouse cursor over a resource bar of a node.
Historical trends of resource occupation will be displayed.

- **Current Job**: indicates the historical trend of CPU or memory utilization of the current job.
- **Others**: indicates the historical trend of CPU or memory utilization of the other jobs.

## GPU resource monitoring

Step 1. On the resource details page, select the **GPU** tab. Then select whether to check GPU or GPU memory utilization of the job in the upper right corner.

Step 2. Place the mouse cursor over a resource bar of a node with a specific GPU.

**Note:** If multiple GPUs are available for a node, resource occupation statistics are displayed for each GPU. In a node pane, the upper left corner displays the node name while the upper right corner displays the GPU ID.

Historical trends of resource occupation will be displayed.

- **GPU Total**: indicates the historical trend of GPU utilization of the current job.
- **GPU Memory Total**: indicates the historical trend of GPU memory utilization of the current job.

---

## VNC management

With the VNC management feature, user can manage the VNC sessions in a cluster. They can perform VNC management through either of the following methods:

- Select **Admin → VNC** to open the VNC page, and click the corresponding open button for the VNC session to view a session. Provide the account number and password if the session is locked.
- Log in to a node in a cluster to manage VNC sessions.
  - Type `vncserver -list` to view the VNC sessions that have been created.
  - Type `vncserver -kill` to delete the VNC sessions that have been created.

  **Notes:**

  - The result from the command-line operations above will be reflected in the VNC management page, but there will be a delay of about 30 seconds.
  - If one user has more than 20 VNC sessions on a node, creating additional sessions on this node may result in failure.

# Chapter 8. Custom templates

LiCO allows users to create custom templates. After the user creates a custom template, it can be published by an administrator and then be utilized by other users.

After a template is published, it cannot be edited. In order to edit the template, an administrator needs to unpublish the custom template first.

## Create a custom template

Step 1. Select **Submit Job** from the left navigation pane.

Step 2. Click **My Templates** in the upper-right corner of the page.
A page is displayed, showing the custom templates of the current user.

Step 3. Click **Create**.

Step 4. Fill in the required information.

a. In the Template Information area, provide the basic template information.
The logo image must be in JPG, PNG, JPEG, or BMP format, with a recommended size of 180*40.

b. In the Template Parameters area, click **Add**, and then fill in the Add Parameter dialog.

**Note:** To filter an image through image tag, select **Container image** in the **Data Type** field, select the target framework in the **Framework** field, and click **Add** to add the container image tags in the **Filter Tags** area.

Step 5. Click **Submit**.

The newly created custom template is displayed if you click **My Templates**.

## Edit a custom template

Step 1. Select **Submit Job** from the left navigation pane.

Step 2. Click **My Templates** in the upper-right corner of the page.

Step 3. Find the template you want to edit, and select **Action → Edit**.

Step 4. Change the template information as required.

Step 5. Click **Submit**.

## Copy a custom template

Step 1. Select **Submit Job** from the left navigation pane.

Step 2. Click **My Templates** in the upper-right corner of the page.

Step 3. Find the template you want to copy, and select **Action → Copy**.
The page for copying the template is displayed.

Step 4. Change the template information as required.

Step 5. Click **Submit**.

## Delete a custom template

Step 1. Select **Submit Job** from the left navigation pane.

Step 2. Click **My Templates** in the upper-right corner of the page.

Step 3. Find the template you want to delete, and select **Action → Delete**.

Step 4. Click **OK**.

## Publish a custom template

Administrator users can publish custom templates on LiCO.

Step 1. Select **Submit Job** from the left navigation pane.

Step 2. Click **My Templates** in the upper-right corner of the page.

Step 3. Find the template you want to publish, and select **Action → Publish**.

Step 4. Fill in the category.

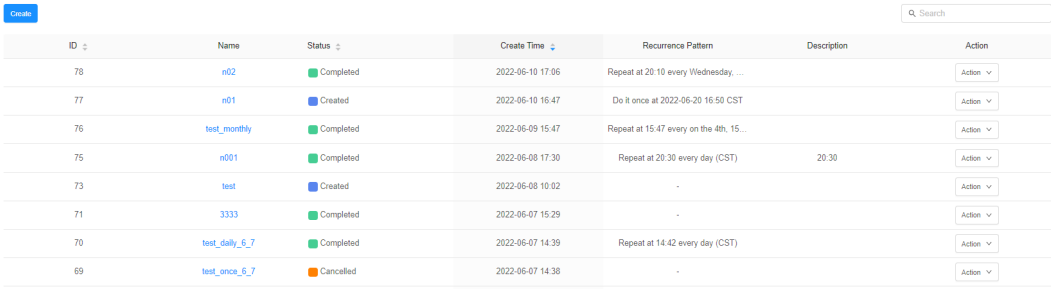
Step 5. Click **Submit**.

The selected template is published.

# Chapter 9. Workflow

Using this function, steps manually executed by users in LiCO can be combined into an automatically executed workflow. Each step can contain one or more jobs, and jobs in a step are executed in a parallel manner (not in any order), while steps are executed in a serial manner.

Select **Workflow** from the left navigation pane to open the Workflow page.




The parameters on the Workflow page are described as follows:

- **Name**: indicates the workflow name, by clicking which you can view the workflow details.
- **Status**: indicates the current workflow status.
  - **created**: indicates that the workflow has been created successfully.
  - **starting**: indicates that the workflow is starting.
  - **running**: indicates that the workflow is running.
  - **cancelling**: indicates that the workflow is being cancelled.
  - **cancelled**: indicates that the workflow has been cancelled.
  - **completed**: indicates that the workflow has completed.
  - **failed**: indicates that the workflow failed to run.
- **Create Time**: indicates the time when the workflow was created.
- **Recurrence Pattern**: indicates the running cycle of workflow.
  - **Once**: indicates that the workflow is only run for once.
  - **Off**: indicates that no timing policy is set for the workflow. Users should turn on the workflow manually.
  - **On**: indicates that the timing policy is set for the workflow. The workflow will be run periodically.
- **Description**: indicates the workflow description.
- **Action**: indicates the available operations on the workflow.
  - If the workflow is in the **created** state, supported operations include **Run**, **Edit**, **Copy**, and **Delete**.
  - If the workflow is in the **starting** or **running** state, supported operations include **Copy** and **Cancel**.
  - If the workflow is in the **cancelling** state, only **Copy** is supported.
  - If the workflow is in the **cancelled**, **completed**, or **failed** state, supported operations include **Rerun**, **Edit**, **Copy**, and **Delete**.

## Create a workflow

Step 1. Select **Workflow** from the left navigation pane.

Step 2. On the Workflow page, click **Create**.

Step 3. Fill in the required information.

- **Name**: self-defined job name
- **Max Submit Jobs**: the maximum number of jobs running and pending in a workflow at a given time
- **Description**: self-defined job description

Step 4. Click **OK**. The page for editing the workflow is displayed.

**Note:** To change the parameters specified in step 3, select **Action → Edit**, and then edit the parameters as required on the Edit Workflow page that is displayed.

Step 5. Click **+Create New Step** for the created workflow, fill in the required parameters on the Create Step page that is displayed, and then click **OK**. (You can perform this step for multiple times to add more steps.)

- **Name**: self-defined step name
- **Description**: self-defined step description

**Note:** To change the parameters specified for a created step, move your cursor over the step name, and then you can edit or delete this step using the edit or delete icon.

Step 6. Click **+Create Job** for the created step, select the required job template and fill in the required job parameters by referring to Chapter 3 “Lenovo-accelerated AI” on page 19, Chapter 6 “Job submission” on page 53, and Chapter 8 “Custom templates” on page 77, and then click **Add to Workflow**. (You can perform this step for multiple times to add more jobs.)

**Notes:**

- To quit the selected job template and use another one, click **Back** on the page for filling in template parameters.
- To discard the job being created, click **Back To Workflow** to return to the page for editing the workflow.
- To perform operations on a created job, click .

Step 7. Select **Action → Run** in the upper right corner of the page for editing the workflow, and then jobs in the current workflow will begin running, and you will be redirected to the Workflow page.

**Note:** To change the current workflow, select **Action → Edit**, and then you can change the workflow parameters on the Edit Workflow page that is displayed.

## Edit a workflow

Step 1. Select **Workflow** from the left navigation pane.

Step 2. Find the workflow you want to edit, and then select **Action → Edit**.

Step 3. Edit the parameters as required.

Step 4. Click **OK**.

## Copy a workflow

Step 1. Select **Workflow** from the left navigation pane.

Step 2. Find the workflow you want to copy, and then select **Action → Copy**.

Step 3. Fill in the required information.

- **Name**: name of the workflow generated after the copy operation
- **Concurrent Jobs**: number of jobs that can concurrently run in a step
- **Description**: self-defined job description
- **Frequency**: this option is displayed when **Appointment Recurrence** is selected. Users can select the running frequency of workflow from the drop-down list.

  **Notes:**

  - When **Once** is selected, the **Recurrence Pattern** will be switched to **Once**, and will be switched to **Off** after the workflow is completed.
  - When **Daily**, **Weekly**, or **Monthly** is selected, the **Recurrence Pattern** will be switched to **On**.
  - **Once**: the workflow will be run in the specified time period only once.
  - **Daily**: the workflow will be run in the specified time period once a day.
  - **Weekly**: the workflow will be run in the specified time period of one or several days in each week.
  - **Monthly**: the workflow will be run in the specified time period of one or several days in each month.
- **Trigger Time**: this option is displayed when **Appointment Recurrence** is selected. Users can set the trigger time for the target workflow.

Step 4. Click **OK**.

## Run a workflow

Step 1. Select **Workflow** from the left navigation pane.

Step 2. Find the workflow you want to run, and then select **Action → Run**.

## Rerun a workflow

Step 1. Select **Workflow** from the left navigation pane.

Step 2. Find the workflow you want to rerun, and then select **Action → Rerun**.

## Cancel a workflow

Step 1. Select **Workflow** from the left navigation pane.

Step 2. Find the running workflow you want to cancel, and then select **Action → Cancel**.

Step 3. Click **Confirm**.

## Delete a workflow

Step 1. Select **Workflow** from the left navigation pane.

Step 2. Find the workflow you want to delete, and then select **Action → Delete**.

Step 3. Click **Confirm**.

# Chapter 10. Reports

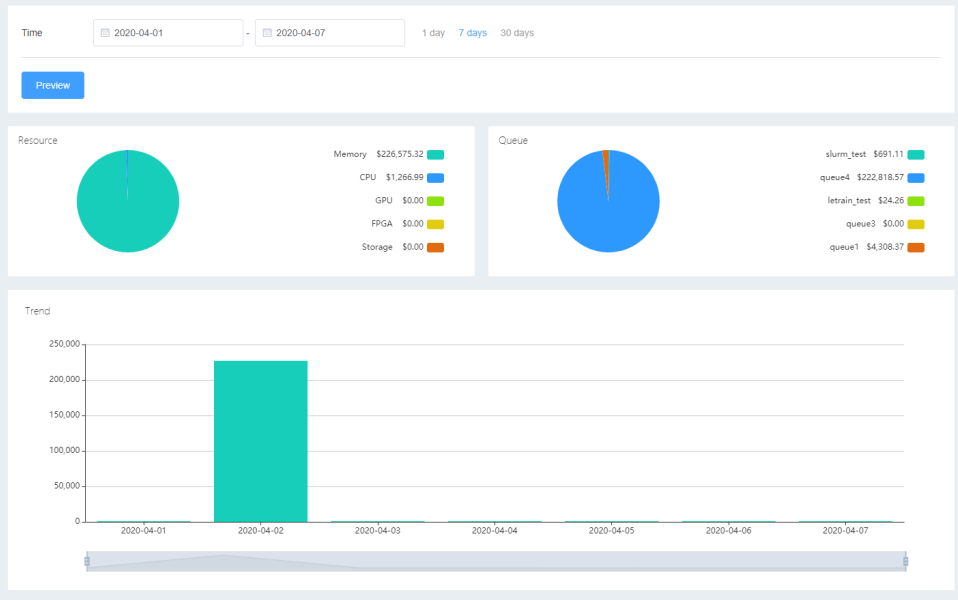
Select **Reports** from the left navigation pane.

**Expense Reports** is supported currently, where the job and storage billing statistics are displayed.

## Expense reports

Select **Reports → Expense Reports** from the left navigation pane.

A page for obtaining reports on expenses is displayed.




The report filters include:

- **Time**: supports pre-defined and self-defined time periods of no longer than one year.

  **Note: 1 day**, **7 days**, or **30 days**: indicates last 1 day, last 7 days, or last 30 days.

The preview function includes:

- **Resource**: displays billing statistics for CPU, memory, storage, and generic resources. Generic resources generally include GPU, FPGA, and so on.
- **Queue**: displays billing statistics for jobs in all queues, excluding storage billing statistics.
- **Trend**: displays daily billing statistics for the time range currently displayed. You can drag the scroll bar horizontally to change the time range displayed.

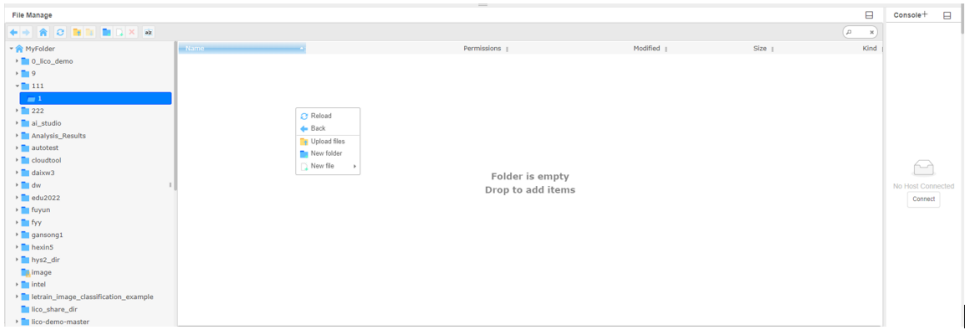
# Chapter 11. How to run a TensorFlow program on LiCO

## Prepare a workspace

Step 1. Log in to the LiCO system as a common user.

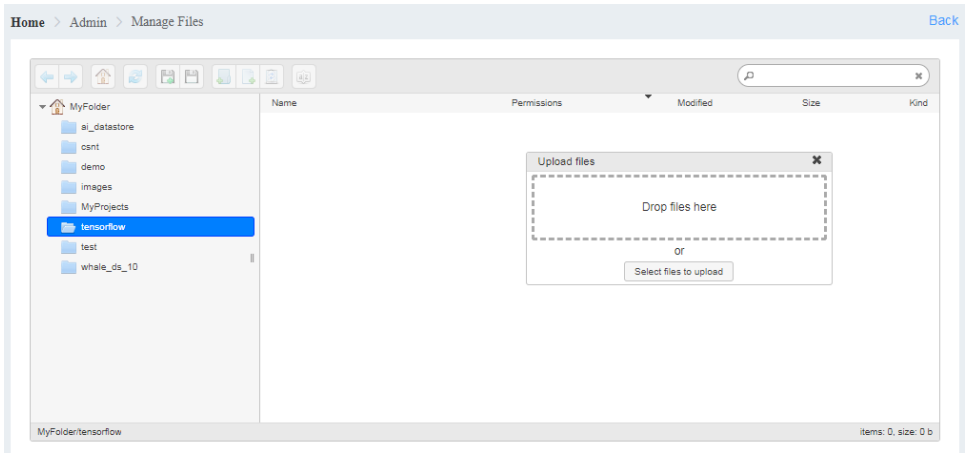
Step 2. Click the system tools icon in the upper-right corner of the home page.

Step 3. Right-click in the blank area of the File Manage page, and select **New folder** from the shortcut menu to create a new directory as a workspace.



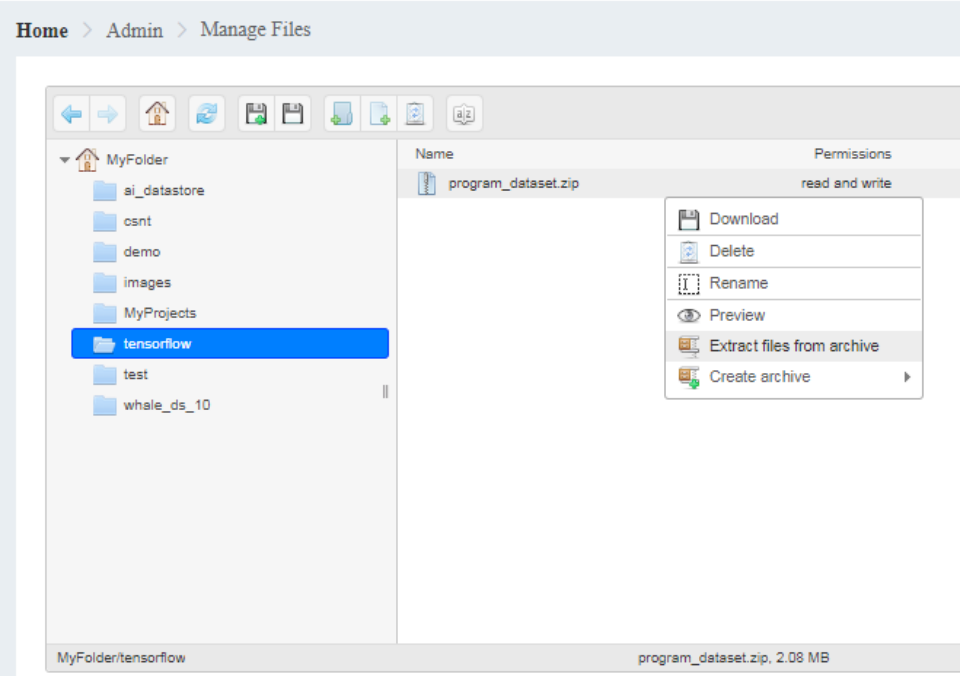

Step 4. Double-click the directory to open it.

Step 5. Right-click in the blank area, and select **Upload files** from the shortcut menu.
The Upload files window is displayed.




Step 6. Click **Select files to upload** to upload the required TensorFlow program and dataset files from the local device.
Archived files are recommended, and ZIP, TAR, and TAR.GZ formats are supported.

Step 7. After the archived file is uploaded, right-click the file and then select **Extract files from archive** from the shortcut menu.




## (Optional) Prepare a container image

**This step is optional. If there is an available container image for your program, skip this step.**

Step 1. Select **Admin → Container Images** from the left navigation pane.

Step 2. Select an appropriate TensorFlow system container image as your base container image.

Step 3. Select **Action → Download** in the same row as the base container image selected.
The Download Container Image dialog is displayed.

Step 4. Select the directory created before as the download path, and click **OK**.

Step 5. Click the system tools icon in the upper-right corner of the home page.

Step 6. On the Console page, click **Connect** or the add icon , log in to one of the login nodes and run the following commands:

```
cd ~/<workspace>

singularity build --sandbox <image_dev_directory>/ <image_filename>

singularity shell -w <image_dev_directory>
```

**Replace <workspace>, <image_dev_directory>, and <image_filename> with actual values.**

Step 7. On the console of the base container image, run commands to install libraries or applications required for your TensorFlow program. For example:

```
pip install pandas
```

Step 8. After you finish preparing the container image, run the following commands to rebuild the container image:

```
exit

singularity build <new_image_filename> <image_dev_directory>/
```

**Replace <new_image_filename> and <image_dev_directory> with actual values.**

Step 9. Select **Admin → Container Images** from the left navigation pane.

Step 10. Click **Import**.
The Import Image dialog is displayed.

Step 11. Fill in the required information.

- For **Framework**, select **Tensorflow**.
- For **Source File**, select the created new container image.

Step 12. Click **OK**.

## Submit a job

Step 1. Select **Submit Job** from the left navigation pane.

Step 2. Select **Industry Standard AI → TensorFlow Multinode → Use**.

Step 3. Fill in the required information.

- For **Workspace**, select the workspace created before.
- For **Container image**, select the private container image uploaded before.
- For **TF program(.py)**, select the TensorFlow python script uploaded into your workspace before.
- If your program uses GPUs, specify **GPU Per Node**. Otherwise, leave it blank.

Step 4. Click **Submit** to submit the job.

## Monitor the job and obtain output files

- If the job is successfully submitted, the system will navigate to the job details page.
- On the job details page, you can view the job log, job file, and launch the TensorBoard for monitoring.
- Select **Action → Browse** in the upper right corner of the job details page.

  A file system browser is displayed and then redirected to the job workspace, where you can download your job output files.

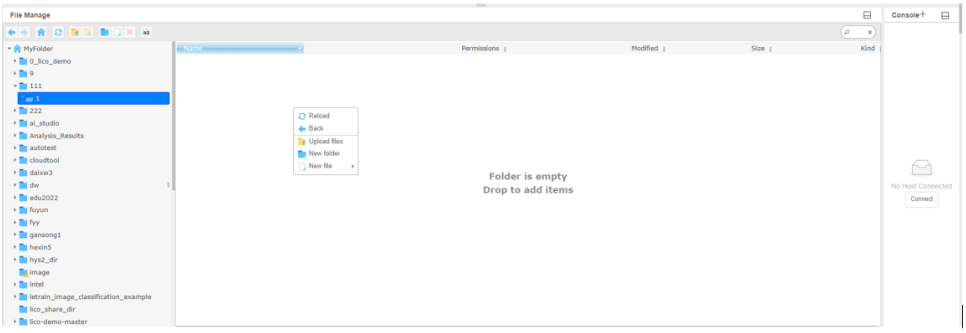
# Chapter 12. How to run a Caffe program on LiCO

## Prepare a workspace

Step 1. Log in to the LiCO system as a common user.

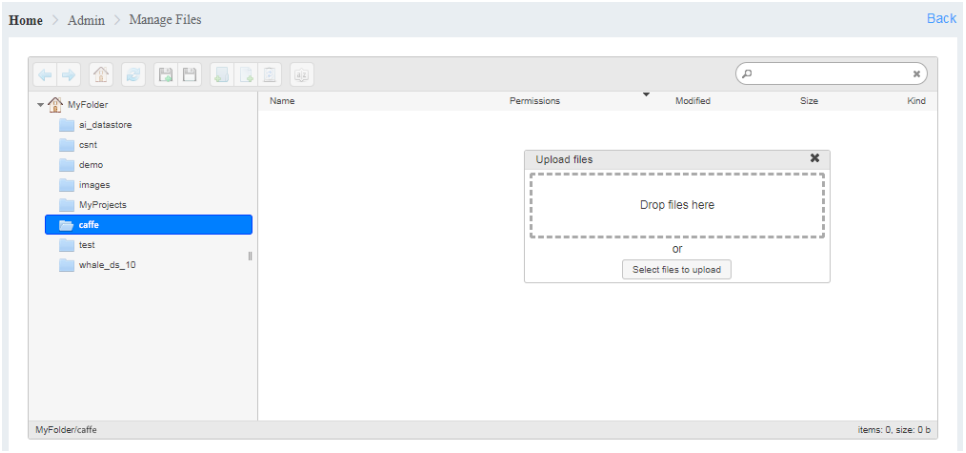
Step 2. Click the system tools icon in the upper-right corner of the home page.

Step 3. Right-click in the blank area of the File Manage page, and select **New folder** from the shortcut menu to create a new directory as a workspace.



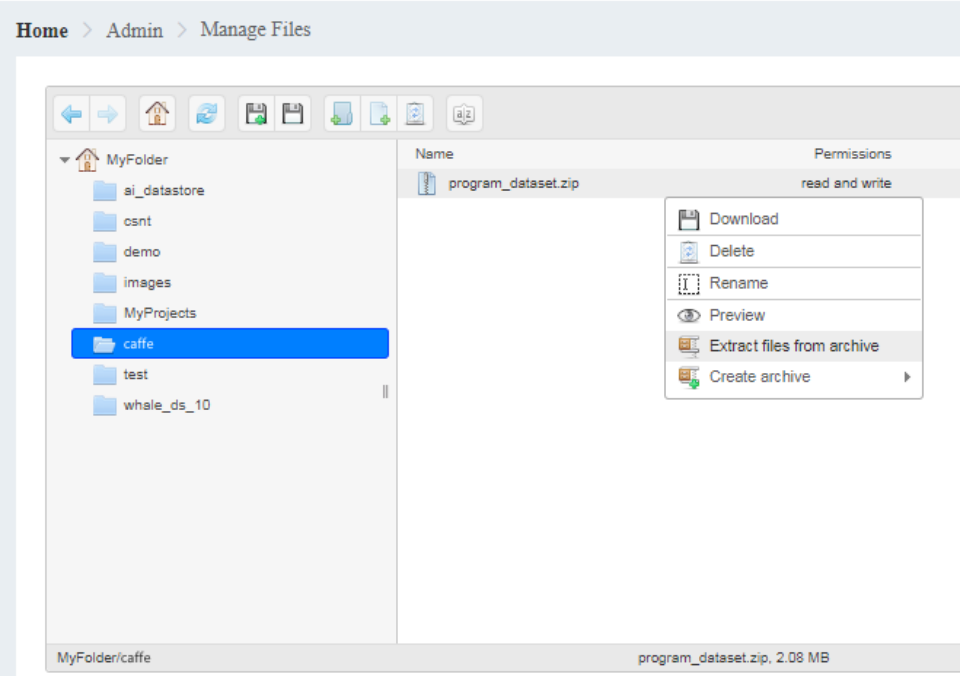

Step 4. Double-click the directory to open it.

Step 5. Right-click in the blank area, and select **Upload files** from the shortcut menu.
The Upload files window is displayed.




Step 6. Click **Select files to upload** to upload the required Caffe program and dataset files from the local device.
Archived files are recommended, and ZIP, TAR, and TAR.GZ formats are supported.

Step 7. After the archived file is uploaded, right-click the file and then select **Extract files from archive** from the shortcut menu.




## (Optional) Prepare a container image

**This step is optional. If there is an available container image for your program, skip this step.**

Step 1. Select **Admin → Container Images** from the left navigation pane.

Step 2. Select an appropriate Caffe system container image as your base container image.

If your program runs on CPUs, select the container image named **caffe-cpu**.

If your program runs on GPUs, select the container image named **caffe-gpu**.

Step 3. Select **Action → Download** in the same row as the base container image selected.
The Download Image dialog is displayed.

Step 4. Select the directory created before as the download path, and click **OK**.

Step 5. Click the system tools icon in the upper-right corner of the home page.

Step 6. On the Console page, click **Connect** or the add icon , log in to one of the login nodes and run the following commands:

```
cd ~/<workspace>

singularity build --sandbox <image_dev_directory>/ <image_filename>

singularity shell -w <image_dev_directory>
```

**Replace <workspace>, <image_dev_directory>, and <image_filename> with actual values.**

Step 7. On the console of the base container image, run commands to install libraries or applications required for your Caffe program. For example:

```
pip install pandas
```

Step 8. After you finish preparing the container image, run the following commands to rebuild the container image:

```
exit

singularity build <new_image_filename> <image_dev_directory>/
```

**Replace <new_image_filename> and <image_dev_directory> with actual values.**

Step 9. Select **Admin → Container Images** from the left navigation pane.

Step 10. Click **Import**.
The Import Image dialog is displayed.

Step 11. Fill in the required information.

- For **Framework**, select **Caffe**.
- For **Source File**, select the created new container image.

Step 12. Click **OK**.

## Submit a job

Step 1. Select **Submit Job** from the left navigation pane.

Step 2. Select **Industry Standard AI → Caffe → Use**.

Step 3. Fill in the required information.

- For **Workspace**, select the workspace created before.
- For **Container image**, select the private container image uploaded before.
- For **Caffe program**, select the Caffe python script uploaded into your workspace before.
- If your program uses GPUs, specify **GPU Per Node**. Otherwise, leave it blank.

Step 4. Click **Submit** to submit the job.

## Monitor the job and obtain output files

- If the job is successfully submitted, the system will navigate to the job details page.

  You can also select **Jobs** from the navigation pane and then click the job name to go to its details page.
- On the job details page, you can view the job log and job file for monitoring.
- Select **Action → Browse** in the upper right corner of the job details page.

  A file system browser is displayed and then redirected to the job workspace, where you can download your job output files.

# Chapter 13. Additional information

## Deploying a publishing image

```
docker pull <image_name>

docker run -p <outer_port>:80 <image_name> [--url=<web_service_url>]
```

- Replace <image_name> with the actual image.
- Replace <outer_port> with the actual port.
- (Optional): Replace <web_service_url> with a custom URL. Its default value is `/api/service`.

**Notes:** Running the commands above is to containerize trained model deployment by generating a deployment service. For details, refer to "Deploy a model" on page 45. Differences between running the commands above and deploying a trained model are as follows:

- Authorization is not required in the request parameters when you run commands to generate a deployment service.
- URLs are different.

## Failed job submissions

In most circumstances failed job submissions in LiCO result from incorrect scheduler service configuration. You can check and resolve problems using the following methods:

- Use an SSH tool to log in to the login node in the system, and re-submit the job with command lines.
  1. Go to the user directory and locate the job file.
  2. Run the command `sbatch jobfile.slurm` to submit the job.
  3. View the error log.

  **Note:** The most common problem is exceeding resource limits. For example, if the system has 80 CPU cores, a job requiring 100 cores will lead to a submission failure.
- Log in to the management node, and run `scontrol show nodes` to check the status of the compute nodes and resources in a cluster. Notify an administrator in the following circumstances:
  - The command returns an empty result, indicating that the node has not been added into the scheduler node.
  - The command returns a result indicating that some nodes are not working.
- Log in to the management node, and run `sinfo` to check queue settings.

  **Note:** You need to notify an administrator if an exception is found.

## Deletion of VNC sessions

If VNC sessions are invisible on the VNC management page, contact an administrator.

If an attempt to delete a VNC session fails, log in to the node associated with the VNC session via SSH, run `vncserver -list` to check the status of the session, and then use the command `vncserver-kill` to delete the session.

About 30 seconds after the deletion, refresh the VNC management page.

## References for Slurm commands

For details about Slurm commands, refer to the following Web site: https://slurm.schedmd.com/quickstart.html.

## Data sources for GPU monitoring

LiCO can only monitor GPUs produced by NVIDIA. Monitoring data (including GPU usage, memory, temperature, and usage status) is obtained through the official NVIDIA API.

To check GPU monitoring data on the node's operating system, run the `nvidia-smi` command.

## Transform an NGC image

Step 1. Register an account on the NGC Web site (https://ngc.nvidia.com).

Step 2. Generate an API KEY as the authentication key to pull the NGC image.

a. Click **Get API Key** on the Registry page.

b. Click **Generate API Key** on the displayed API Key page to get an API key.

Step 3. Transform the NGC image with Singularity.

a. Run the following command to check the Singularity version:

```
singularity --version
```

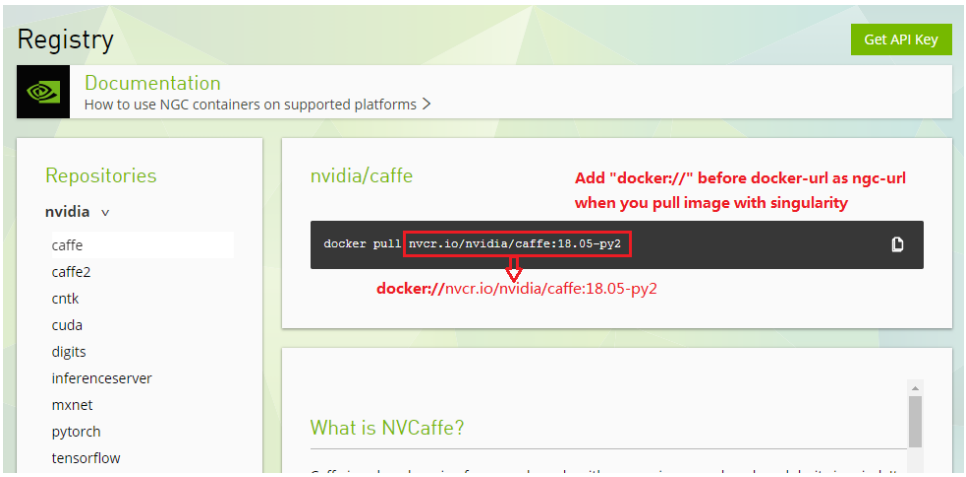
**Note:** Ensure that the Singularity version is later than or equal to 3.1.0.

b. Run the following commands to transform the NGC image to a Singularity container image:

```
# Set the Singularity environment variable

export SINGULARITY_DOCKER_USERNAME='$oauthtoken'

export SINGULARITY_DOCKER_PASSWORD=<API_KEY>

export SINGULARITY_PULLFOLDER=<IMAGE_FOLDER>

# Transform image

singularity pull --name <IMAGE _FILENAME> <NVIDIA-DOCKER-URL>
```

**Notes:**

- Replace <API_KEY> with your specific API KEY.
- Replace <IMAGE_FOLDER> with a specific folder to save the Singularity container image.
- Replace <IMAGE_ FILENAME> with a specific name for the Singularity container image. For example: ngc-caffe.image.
- Replace <NVIDIA-DOCKER-URL> with a specific URL of the NVIDIA docker. You can get the NVIDIA docker URL from https://ngc.nvidia.com/registry (shown below). Please select the latest version of the NGC docker image.

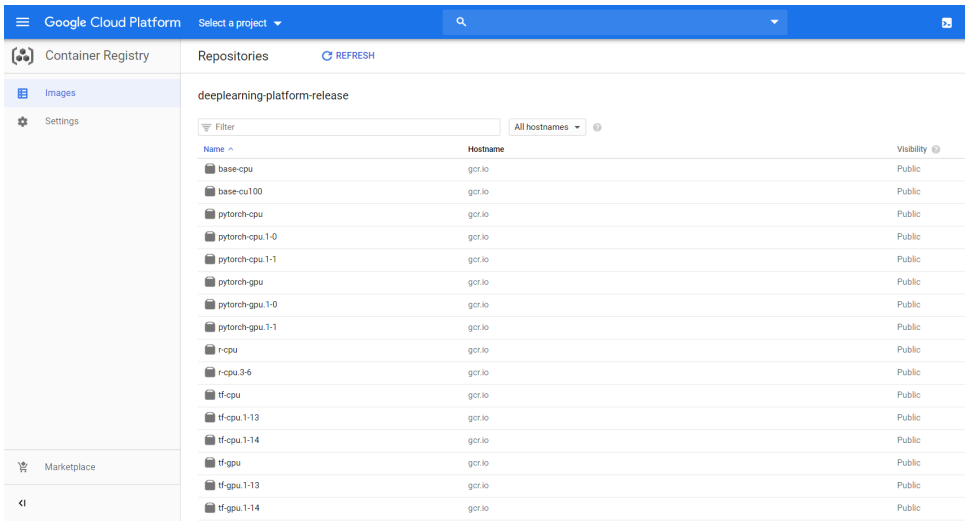



If the commands above are executed successfully, you can see a Singularity container image <IMAGE_ FILENAME> under <IMAGE_FOLDER>.

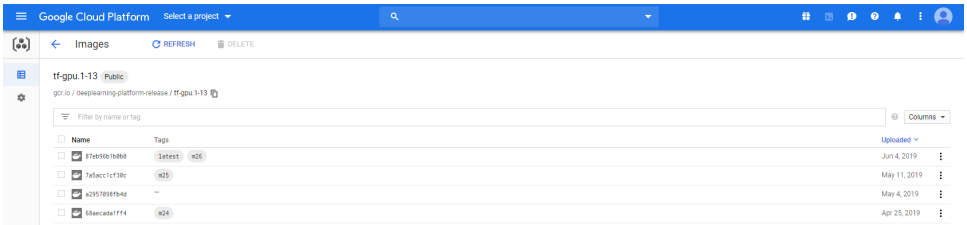
Step 4. Upload the container image by referring to "Import a container image" on page 10.

## Transform a Google deep learning container

Step 1. Obtain the Google deep learning container URL.
Google provides pre-configured and optimized deep learning containers that can be used within LiCO. To view available container images, visit https://console.cloud.google.com/gcr/images/deeplearning-platform-release?project=deeplearning-platform-release.

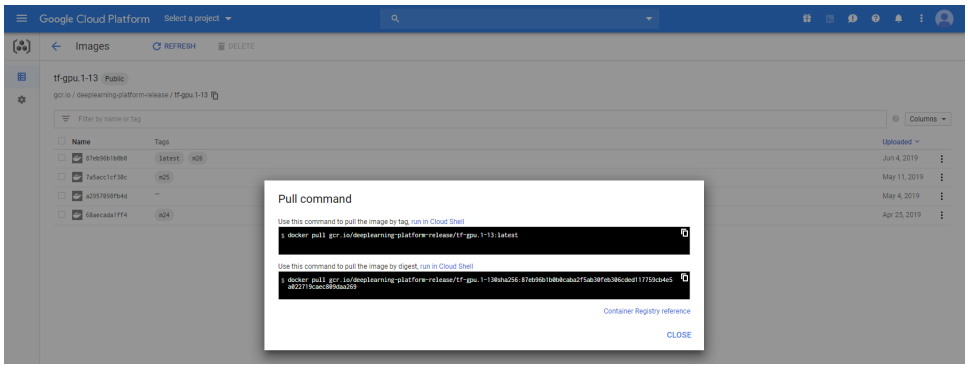



a. Select a container for your DL framework, for example, GPU-enabled Tensorflow 13.



b. Find the container image version you want to use (the one with the “latest” tag is recommended), click ⋮, and select the **Show Pull Command** option.

c. Copy the last part of the command shown on the displayed page, that is, *gcr.io/deeplearning-platform-release/tf-gpu.1-13:latest* in this example.




Step 2. Prepare a recipe file.
Create a new file named recipe.def, using the URL reference obtained in step 1:

```
Bootstrap: docker

From: gcr.io/deeplearning-platform-release/tf-gpu.1-13:latest

%post

chmod 755 /root
```

Step 3. Build a singularity container image from the recipe file.
Build a container image on the server with singularity by running the following command as a **root** user:

```
$ singularity build tf-gpu.1-13.image recipe.def
```

A container named tf-gpu-1.13.img is created.

Step 4. Upload the container image by referring to “Import a container image” on page 10.

**Note:** Probable issue: If you encounter a crash problem during job running, try to reduce the CPU cores when submitting the job.

## Publishing issues troubleshooting

When you create a Docker publishing task, the following error messages may be displayed:

```
ERRO error unmounting /var/lib/containers/storage/overlay/
c06ee04b424ea968a62d259ec55fc19091cfc97a6e6ee8c1ec61c5ddb7b5bda9/merged: invalid argument

Error initializing source containers-storage:letrain-working-ctracting layer
"8ed1111e10c9061ca4cdb8891217b412aadd9cf7d022e753ff278b1a1eda06dc": cannot mount layer, mount label too
large 5143
```

In this case, visit https://github.com/containers/buildah/blob/master/install.md#configuration-files to upgrade or configure Buildah.

## Known issues

- For deep learning, if training loss is NAN, the training process is a failure. In this case, continuing training or using it for prediction may cause unexpected results, such as a program crash or no prediction result. If the NAN loss value occurs in the training process, you are recommended to cancel the training job, adjust the parameters (such as reducing the learning rate), and then submit a new training job.
- Due to model implementation reasons, for a multi-node training job, the final step in the log may be inconsistent with the maximum step set on the LiCO portal, but the actual training step for this job is the same as the user setting.
- When an instance segmentation job is running on multiple nodes, it may be hung and get the NAN loss value. This is a known issue and should be fixed in future releases. Check the LiCO product page or contact sales personnel to get related information.
- Jobs may get stuck when training continues based on an existing training checkpoint (stored in Train Directory). In this case, delete the Train Directory of the original training model before continuing the training.
- If you have a pop-up blocker enabled, some LiCO feature pages may not be launched. In this case, add these pages to the white list.
- For OpenHPC V2.3 and later versions, **pmix** is not supported in Slurm.

## Notices and trademarks

### Notices

Lenovo may not offer the products, services, or features discussed in this document in all countries. Consult your local Lenovo representative for information on the products and services currently available in your area. Any reference to a Lenovo product, program, or service is not intended to state or imply that only that Lenovo product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any Lenovo intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any other product, program, or service.

Lenovo may have patents or pending patent programs covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

*Lenovo (United States), Inc.*
*8001 Development Drive*
*Morrisville, NC 27560*
*U.S.A.*
*Attention: Lenovo Director of Licensing*

LENOVO PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

Changes are made periodically to the information herein; these changes will be incorporated in new editions of the publication. To provide better service, Lenovo reserves the right to improve and/or modify the products and software programs described in the manuals included with your computer, and the content of the manual, at any time without additional notice.

The software interface and function and hardware configuration described in the manuals included with your computer might not match exactly the actual configuration of the computer that you purchase. For the configuration of the product, refer to the related contract (if any) or product packing list, or consult the distributor for the product sales. Lenovo may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

The products described in this document are not intended for use in implantation or other life support applications where malfunction may result in injury or death to persons. The information contained in this document does not affect or change Lenovo product specifications or warranties. Nothing in this document shall operate as an express or implied license or indemnity under the intellectual property rights of Lenovo or third parties. All information contained in this document was obtained in specific environments and is presented as an illustration. The result obtained in other operating environments may vary.

Lenovo may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any references in this publication to non-Lenovo Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this Lenovo product, and use of those Web sites is at your own risk.

Any performance data contained herein was determined in a controlled environment. Therefore, the result obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.



For the latest information or any questions or comments, contact or visit the Lenovo Web site:

https://support.lenovo.com

**Trademarks**