

# Alibaba Cloud

## Elastic GPU Service Best Practices

Document Version: 20220713

# Legal disclaimer

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

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.
2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company or individual in any form or by any means without the prior written consent of Alibaba Cloud.
3. The content of this document may be changed because of product version upgrade, adjustment, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and an updated version of this document will be released through Alibaba Cloud-authorized channels from time to time. You should pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.
4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides this document based on the "status quo", "being defective", and "existing functions" of its products and services. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not take legal responsibility for any errors or lost profits incurred by any organization, company, or individual arising from download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, take responsibility for any indirect, consequential, punitive, contingent, special, or punitive damages, including lost profits arising from the use or trust in this document (even if Alibaba Cloud has been notified of the possibility of such a loss).
5. By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.
6. Please directly contact Alibaba Cloud for any errors of this document.

# Document conventions

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

# Table of Contents

1. Configure a license server for a GRID driver .....	05
1.1. Configure a Linux license server .....	05
1.2. Configure a Windows license server .....	10
2. Deploy an NGC environment on a GPU-accelerated instance .....	14
3. Use RAPIDS to accelerate machine learning tasks on a GPU-ac... ..	18

# 1. Configure a license server for a GRID driver

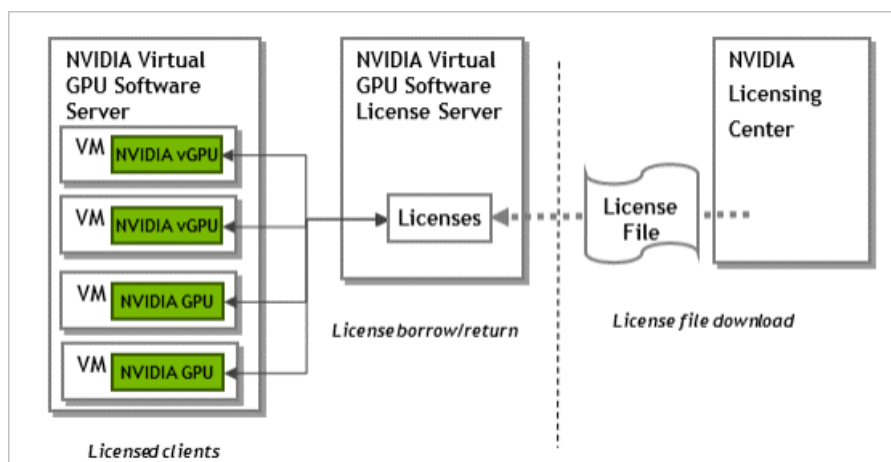
## 1.1. Configure a Linux license server

Before you use an NVIDIA GRID driver, you must apply for the driver license and install the license on a license server. This topic describes how to configure a Linux license server.

### Context

In this example, Ubuntu 18.04 is used. If your instance runs another type of Linux other than Ubuntu, for example, CentOS, you must change specific commands that are provided in this topic based on your business requirements. For more information, see [Virtual GPU software license server documentation](#).

The following figure shows the licensing architecture of an NVIDIA GRID driver.



You must obtain the license file of a GRID driver from the NVIDIA Licensing Portal and install the file on a license server that you configured. Then, you must connect your GPU-accelerated or vGPU-accelerated instance to the license server over the Internet and activate the license of the GRID driver.

### Step 1: Create an ECS instance

Create a Linux Elastic Compute Service (ECS) instance that works as a license server. For more information, see [Create an instance by using the wizard](#).

When you create the instance, you must configure parameters based on the descriptions that are provided in the following table. You can configure the parameters that are not provided in the table based on your business requirements.

Parameter	Description
Instance specifications	Select an ECS instance whose specifications are higher than the specifications of 2 vCPUs and 4 GB memory. If you require a large number of licenses, we recommend that you select an ECS instance whose specifications are higher than the specifications of 4 vCPUs and 16 GB memory to ensure high performance and stability.
Image	Select a Linux image. In this example, an Ubuntu 18.04 image is used.

Parameter	Description
Storage	Select a system disk whose size is larger than 40 GiB.
Bandwidth	We recommend that you select the pay-by-data-transfer metering method and set the peak bandwidth to 100 Mbit/s.

## Step 2: Install JRE

1. Connect to the ECS instance that works as the license server.

For more information, see [Connect to a Linux instance by using a password or key](#).

2. Run the following command and check whether Java Runtime Environment (JRE) is installed on the instance:


```
java -version
```

If the returned information about the Java version is similar to the following command output, JRE is installed. If the returned information about the Java version is different from the following command output, perform the next operation.

```
java version "1.7.0_51"  
OpenJDK Runtime Environment (rhel-2.4.5.5.el7-x86_64 u51-b31)  
OpenJDK 64-Bit Server VM (build 24.51-b03, mixed mode)
```

3. Run the following command to install OpenJDK:

```
sudo apt install default-jdk
```

 **Note** If the message `Unable to locate package default-jdk` appears, run the `apt update` command before you run the `sudo apt install default-jdk` command.

If the returned information is similar to the following command output, OpenJDK is installed.

```
...  
Running hooks in /etc/ca-certificates/update.d...  
done.  
done.
```

## Step 3: Install and run Apache Tomcat

1. Run the following command to install the Apache Tomcat package by using your package manager of the Linux distribution:

```
sudo apt install tomcat8
```


2. After you install Apache Tomcat, run the following command to enable automatic startup for Apache Tomcat:

```
sudo systemctl enable tomcat8.service
```

3. Run the following command to start Apache Tomcat:

```
sudo systemctl start tomcat8.service
```

4. Open `http://localhost:8080` in your web browser and check whether Apache Tomcat is available.

 **Note** Replace `localhost` in the URL with the public IP address of your ECS instance.

If the message that is shown in the following figure appears, Apache Tomcat is installed.

#### It works !

If you're seeing this page via a web browser, it means you've setup Tomcat successfully. Congratulations!

This is the default Tomcat home page. It can be found on the local filesystem at: `/usr/lib/tomcat8/webapps/ROOT/index.html`

Tomcat8 veterans might be pleased to learn that this system instance of Tomcat is installed with `CATALINA_HOME` in `/usr/share/tomcat8` and `CATALINA_BASE` in `/usr/lib/tomcat8`, following the rules from `/usr/share/doc/tomcat8-common/README.txt.gz`.

You might consider installing the following packages, if you haven't already done so:

**tomcat8-docs**: This package installs a web application that allows to browse the Tomcat 8 documentation locally. Once installed, you can access it by clicking [here](#).

**tomcat8-examples**: This package installs a web application that allows to access the Tomcat 8 Servlet and JSP examples. Once installed, you can access it by clicking [here](#).

**tomcat8-admin**: This package installs two web applications that can help managing this Tomcat instance. Once installed, you can access the [manager webapp](#) and the [host-manager webapp](#).


NOTE: For security reasons, using the manager webapp is restricted to users with role "manager-gui". The host-manager webapp is restricted to users with role "admin-gui". Users are defined in `/etc/tomcat8/tomcat-users.xml`.

## Step 4: Install the license server software

Before you install the license server software, make sure that you have obtained the `setup.bin` file that contains the installation package of the license server software or the `.zip` file that contains the `setup.bin` file.

1. Download the `setup.bin` file that contains the installation package or decompress the `.zip` file.
  - o If you obtained the installation package of the license server software from the NVIDIA official website, run the following command to decompress the package:

```
unzip The name of the installation package.zip
```

 **Note** Replace *The name of the installation package* in the command with the name of the installation package that you obtained.

- o If you obtained the download URL of the installation package from Alibaba Cloud, run the following command to download the installation package:

```
wget https://grid-9-4-zyy.oss-cn-hangzhou.aliyuncs.com/setup.bin
```

2. Run the following command to grant execute permissions on the installation package:

```
chmod +x setup.bin
```

3. Install the license server software.

- i. Run the following command to install the license server software as the root user:

```
sudo ./setup.bin -i console
```

- ii. In the **Introduction** section, press Enter.

```
=====
Introduction
-----

InstallAnywhere will guide you through the installation of License Server.

It is strongly recommended that you quit all programs before continuing with
this installation.

Respond to each prompt to proceed to the next step in the installation.  If
you want to change something on a previous step, type 'back'.

You may cancel this installation at any time by typing 'quit'.

PRESS <ENTER> TO CONTINUE: |
```

- iii. In the **License Agreement** section, press Enter each time you are prompted to page through the license agreement.

When you reach the end of the agreement, you are prompted to accept the terms of the agreement. To accept the terms, enter *y* and press Enter.

```
DO YOU ACCEPT THE TERMS OF THIS LICENSE AGREEMENT? (Y/N): Y |
```

- iv. In the **Choose Install Folder** section, press Enter to use the default installation directory.
- v. In the **Choose Local Tomcat Server Path** section, enter the local directory of Apache Tomcat. The default directory is */var/lib/tomcat* the version number of Apache Tomcat. Example: */var/lib/tomcat8*.
- vi. In the **Choose Firewall Options** section, select the ports that you want to open in the firewall and press Enter. We recommend that you keep the default settings.
- vii. In the **Pre-Installation Summary** section, confirm your settings and press Enter. The system starts to install the license server software.
- viii. In the **Install Complete** section, press Enter. The license server software is installed.

## Step 5: Create the license server on the NVIDIA Licensing Portal


1. Log on to the [NVIDIA Licensing Portal](#) by using the email address that you have used to apply for the driver license.
2. On the **Dashboard** page, click **CREATE LICENSE SERVER** in the **License Servers** section.
3. In the **Create License Server** dialog box, configure parameters and click **CREATE LICENSE SERVER**.

The following table describes the parameters that you must configure.

Parameter	Description
Server Name	Specify a custom name for the license server.
MAC Address	Enter the MAC address of the ECS instance that works as the license server. To query the MAC address, log on to the instance and run the <code>ipconfig -a</code> command.
Feature	Select features based on your business requirements, enter the number of licenses that you want to add to the server, and then click <b>ADD</b> .


4. After you create the license server on the NVIDIA Licensing Portal, go to the **License Servers**



section and click the  icon to download the license file.

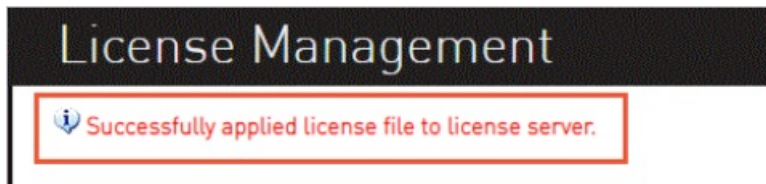
## Step 6: Upload the license file

1. Open `http://localhost:8080/licserver` in your web browser. You are redirected to the license management page.

 **Note** Replace `localhost` in the URL with the public IP address of your ECS instance.

2. In the **License Server** section of the left-side navigation pane, click **License Management**.
3. On the **License Management** page, click **Choose File** on the right of **Upload license file (.bin file)**, select the license file from your computer, and then click **Open**.
4. Click **Upload**.

If the message that is shown in the following figure appears, the license file is uploaded.



To view the number of licenses on the server and the usage details of the licenses, click **License Feature Usage** in the **License Server** section of the left-side navigation pane.

## Step 7: Test network connectivity and access to the license server

In this example, a Windows vGPU-accelerated instance that belongs to the vgn6i instance family is used. If you have a GPU-accelerated instance, you can use the instance.

1. Create a GPU-accelerated instance.

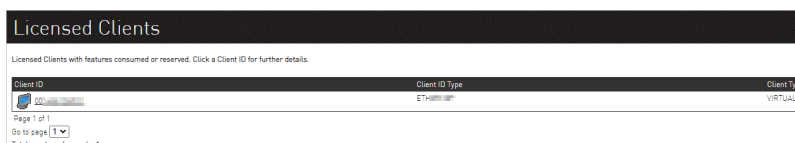
For more information, see [Create a GPU-accelerated instance that is not configured with a driver](#).

In this example, a vGPU-accelerated instance that belongs to the vgn6i instance family and whose image is of Windows Server 2019 Datacenter 64-bit (English) is used.

2. Install the GRID driver, add the license server, and then activate the license. For more information, see [Install a GRID driver on a Windows GPU-accelerated instance](#).

When you activate the license, enter the public IP address of the ECS instance that works as the license server and set the port number to 7070.

3. Open `http://localhost:8080/licserver` in your local web browser. You are redirected to the license management page. In the **License Server** section of the left-side navigation pane, click **Licensed Clients**. On the Client Details page, check whether the information about the vGPU-accelerated instance appears. If the information appears, your vGPU-accelerated instance that works as a virtual machine of the NVIDIA vGPU software client is using the license.



You can click the client ID to view the details of the vGPU-accelerated instance.



## Context

The diagram illustrates the NVIDIA vGPU licensing architecture. It is divided into three main sections by a vertical dashed line:

- Left Section (Licensed clients):** Contains an "NVIDIA Virtual GPU Software Server" box. Inside this box are four "VM" boxes. The first two are labeled "NVIDIA vGPU" (green), and the last two are labeled "NVIDIA GPU" (green). Arrows point from these VMs to the "Licenses" box in the middle section.
- Middle Section:** Contains an "NVIDIA Virtual GPU Software License Server" box. Inside is a "Licenses" box. Below this box is the label "License borrow/return".
- Right Section:** Contains an "NVIDIA Licensing Center" box. Below it is a "License File" box (represented by a document icon). Below this box is the label "License file download".

Arrows indicate the flow of data:

- A dashed arrow points from the "License File" to the "Licenses" box.
- A solid arrow points from the "Licenses" box to the "NVIDIA Virtual GPU Software Server" box.
- Arrows point from each of the four VMs in the "NVIDIA Virtual GPU Software Server" box to the "Licenses" box.

## Step 1: Create an ECS instance

When you create the instance, you must configure parameters based on the descriptions that are provided in the following table. You can configure the parameters that are not provided in the table based on your business requirements.


10

Parameter	Description
Storage	Select a system disk whose size is larger than 40 GiB.
Bandwidth	We recommend that you select the pay-by-data-transfer metering method and set the peak bandwidth to 100 Mbit/s.

## Step 2: Install JRE

1. Connect to the ECS instance that works as the license server.  
For more information, see [Connect to a Windows instance by using a password or key](#).
2. Go to [the ojdkbuild page](#) on GitHub to download the installation package of OpenJDK Java Runtime Environment (JRE).
3. Install JRE.
4. Create the JAVA\_HOME system variable and set the system variable to the absolute path to the *jre* folder of your JRE.

For example, set JAVA\_HOME to *C:\Program Files\ojdkbuild\java-1.8.0-openjdk-1.8.0.201-1\jre*.

 **Note** Make sure that the path does not contain trailing characters, such as backslashes (\) or spaces.

5. Check whether the Path system variable contains the absolute path to the java.exe file.  
In most cases, the path is added when JRE is installed.

## Step 3: Install the license server software

Before you install the license server software, make sure that you have obtained the installation package of the license server software.

1. Decompress the .zip file of the license server software and run setup.exe.
2. On the **Introduction** page, click **Next**.
3. On the **License Agreement** page, select **I accept the terms of the License Agreement** and click **Next**.
4. On the **Apache License Agreement** page, select **I accept the terms of the License Agreement** and click **Next**.
5. On the **Choose Install Folder** page, specify a folder in which you want the license server software to be installed and click **Next**.
6. On the **Choose Firewall Options** page, keep the default setting **License server (port 7070)** and click **Next**.
7. On the **Pre-Installation Summary** page, confirm your settings and click **Install**.
8. After the license server software is installed, click **Done**.


## Step 4: Create the license server on the NVIDIA Licensing Portal

1. Log on to the [NVIDIA Licensing Portal](#) by using the email address that you have used to apply for the driver license.
2. On the **Dashboard** page, click **CREATE LICENSE SERVER** in the **License Servers** section.

3. In the **Create License Server** dialog box, configure parameters and click **CREATE LICENSE SERVER**.


The following table describes the parameters that you must configure.

Parameter	Description
Server Name	Specify a custom name for the license server.
MAC Address	Enter the MAC address of the ECS instance that works as the license server. To query the MAC address, log on to the instance and run the <code>ipconfig -a</code> command.
Feature	Select features based on your business requirements, enter the number of licenses that you want to add to the server, and then click <b>ADD</b> .

4. After you create the license server on the NVIDIA Licensing Portal, go to the **License Servers** section and click the  icon to download the license file.

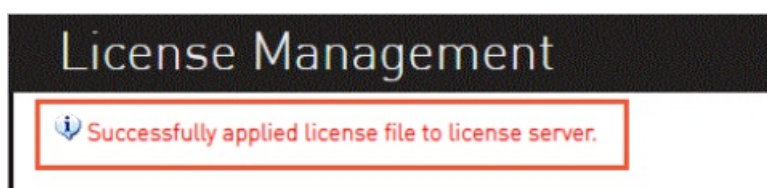
## Step 5: Upload the license file

1. Open `http://localhost:8080/licserver` in your web browser. You are redirected to the license management page.

 **Note** Replace `localhost` in the URL with the public IP address of your ECS instance.

2. In the **License Server** section of the left-side navigation pane, click **License Management**.
3. On the **License Management** page, click **Choose File** on the right of **Upload license file (.bin file)**, select the license file from your computer, and then click **Open**.
4. Click **Upload**.

If the message that is shown in the following figure appears, the license file is uploaded.



To view the number of licenses on the server and the usage details of the licenses, click **License Feature Usage** in the **License Server** section of the left-side navigation pane.

## Step 6: Test the network connectivity and access to the license server

In this example, a Windows vGPU-accelerated instance that belongs to the vgn6i instance family is used. If you have a GPU-accelerated instance, you can use the instance.

1. Create a GPU-accelerated instance.

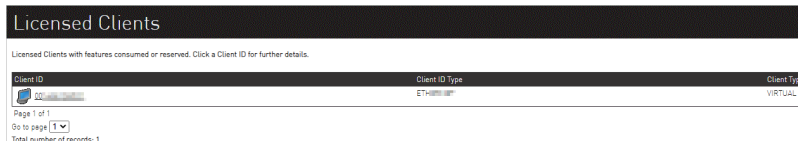
For more information, see [Create a GPU-accelerated instance that is not configured with a driver](#).

In this example, a vGPU-accelerated instance that belongs to the vgn6i instance family and whose image is of Windows Server 2019 Datacenter 64-bit (English) is used.

2. Install the GRID driver, add the license server, and then activate the license. For more information, see [Install a GRID driver on a Windows GPU-accelerated instance](#).

When you activate the license, enter the public IP address of the ECS instance that works as the license server and set the port number to 7070.


3. Open <http://localhost:8080/licserver> in your local web browser. You are redirected to the license management page. In the License Server section of the left-side navigation pane, click **Licensed Clients**. On the Client Details page, check whether the information about the vGPU-accelerated instance appears. If the information appears, your vGPU-accelerated instance that works as a virtual machine of the NVIDIA vGPU software client is using the license.



Client ID	Client ID Type	Client Type
0F-0011-0000	ETHERNET	VIRTUAL

Page 1 of 1  
Go to page 1  
Total number of records: 1

You can click the client ID to view the details of the vGPU-accelerated instance.



Feature Name	Version	Used	Expiry	Vendor String
QuadroVirtual-0000	5.0	1	2022-01-01 00:00:00	QuadroVirtual-0000

## 2. Deploy an NGC environment on a GPU-accelerated instance

This topic describes how to deploy an NVIDIA GPU Cloud (NGC) environment on a GPU-accelerated instance. In this topic, a TensorFlow deep learning framework is used as an example.

### Prerequisites

- An NGC account is created from the [NGC website](#).
- The NGC API key is obtained from the [NGC website](#) and saved to your computer. When you log on to an NGC container environment, the system verifies your NGC API key.

### Context

NGC is a deep learning ecosystem that is developed by NVIDIA. NGC allows developers to access software stacks for free and use the stacks to build development environments for deep learning.

Alibaba Cloud provides instances of the gn5 instance family that are configured with NGC. Alibaba Cloud also provides NGC container images that are optimized for NVIDIA Pascal GPUs in Alibaba Cloud Marketplace. The NGC container images allow developers to quickly deploy NGC container environments and access optimized deep learning frameworks. This way, you can develop and deploy services, and pre-install development environments in an efficient manner. The NGC container images also support optimized algorithm frameworks and real-time updates.

The [NGC website](#) provides various image versions for mainstream deep learning frameworks, such as Caffe, Caffe2, Microsoft Cognitive Toolkit (CNTK), MXNet, TensorFlow, Theano, and Torch. You can select an image based on your business requirements to deploy an environment.

You can deploy an NGC environment on an instance that belongs to one of the following instance families:

- gn4, gn5, gn5i, gn6v, gn6i, and gn6e
- ebmg5i, ebmg6i, ebmg6v, and ebmg6e

The following information describes how to create a GPU-accelerated instance and deploy an NGC environment on the instance. In this example, a GPU-accelerated instance of the gn5 instance family is used.

### Procedure

1. Create a GPU-accelerated instance of the gn5 instance family. For more information, see [Create an instance by using the wizard](#).

When you configure parameters for the instance, take note of the following items:

- **Region:** Select only one of the following regions: China (Qingdao), China (Beijing), China (Hohhot), China (Hangzhou), China (Shanghai), China (Shenzhen), China (Hong Kong), Singapore (Singapore), Australia (Sydney), US (Silicon Valley), US (Virginia), and Germany (Frankfurt).
- **Instance:** Select an instance of the gn5 instance family.
- **Image:** Click **Marketplace Image** and click **Select from Alibaba Cloud Marketplace (including operating system)**. In the dialog box that appears, find **NVIDIA GPU Cloud Virtual Machine Image** and click **Use**.
- **Public bandwidth:** Select **Assign Public IPv4 Address**.

**Note** If you do not select Assign Public IPv4 Address, you can bind an elastic IP address (EIP) to the instance after the instance is created.

- **Security Group:** Select a security group. You must enable TCP port 22 for the security group. If you need your instance to support HTTPS or Deep Learning GPU Training System (DIGITS 6), you must enable TCP port 443 for HTTPS or TCP port 5000 for DIGITS 6.

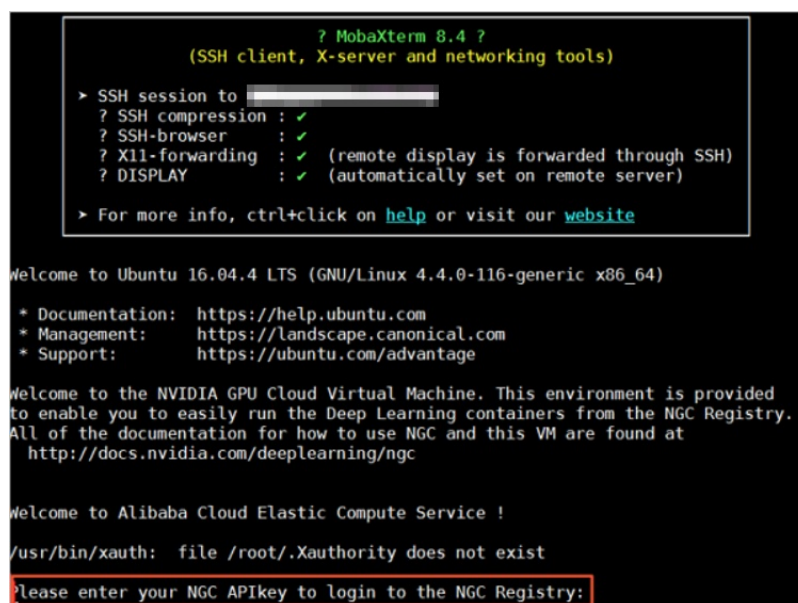
After the GPU-accelerated instance is created, log on to the [ECS console](#) to obtain the public IP address of the instance.

## 2. Connect to the GPU-accelerated instance.

You can use one of the following logon credentials that you selected when you created the instance to connect to the instance:

- Connect to the GPU-accelerated instance by using a password. For more information, see [Connect to a Linux instance by using a password](#).
- Connect to the GPU-accelerated instance by using an SSH key pair. For more information, see [Connect to a Linux instance by using an SSH key pair](#)

## 3. Enter the NGC API Key that you obtained from the NGC website. Then, press the Enter key to log on to the NGC container environment.



```
? MobaXterm 8.4 ?
(SSSH client, X-server and networking tools)

> SSH session to [redacted]
? SSH compression : ✓
? SSH-browser      : ✓
? X11-forwarding   : ✓ (remote display is forwarded through SSH)
? DISPLAY          : ✓ (automatically set on remote server)

> For more info, ctrl+click on help or visit our website

Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-116-generic x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:        https://ubuntu.com/advantage

Welcome to the NVIDIA GPU Cloud Virtual Machine. This environment is provided
to enable you to easily run the Deep Learning containers from the NGC Registry.
All of the documentation for how to use NGC and this VM are found at
http://docs.nvidia.com/deeplearning/ngc

Welcome to Alibaba Cloud Elastic Compute Service !

/usr/bin/xauth: file /root/.Xauthority does not exist

Please enter your NGC APIkey to login to the NGC Registry:
```

## 4. Run the `nvidia-smi` command.

You can view information about the GPU that the instance uses, such as the GPU model and the driver version. The following figure shows the information about the GPU.



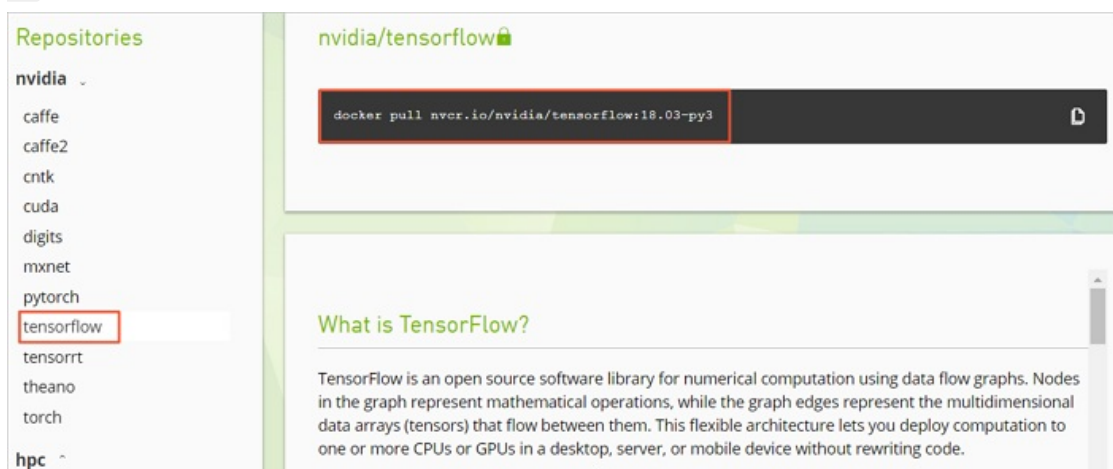
```
root@ ~ # nvidia-smi
Thu Mar 29 20:50:01 2018

+-----+
| NVIDIA-SMI 384.111                Driver Version: 384.111 |
+-----+-----+
| GPU Name Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|     Memory-Usage | GPU-Util  Compute M. |
+-----+-----+
| 0   Tesla P100-PCIE... Off      | 00000000:00:08:0 Off |             0         |
| N/A   29C   P0      27W / 250W | 0MiB / 16276MiB |           0%      Default |
+-----+-----+

Processes:
GPU      PID    Type   Process name                      GPU Memory
Usage
+-----+
| No running processes found |
+-----+
```

## 5. Build a TensorFlow deep learning framework.

- i. Log on to the [NGC website](#), go to the TensorFlow image page, and then copy the `docker pull` command.



- ii. Download the TensorFlow image.

```
docker pull nvcr.io/nvidia/tensorflow:18.03-py3
```

- iii. View the downloaded image.

```
docker image ls
```

- iv. Run the container to deploy the TensorFlow development environment.

```
nvidia-docker run --rm -it nvcr.io/nvidia/tensorflow:18.03-py3
```

```
root@ ~ # nvidia-docker run --rm -it nvcr.io/nvidia/tensorflow:18.03-py3

=====
== TensorFlow ==
=====

NVIDIA Release 18.03 (build 349854)

Container image Copyright (c) 2018, NVIDIA CORPORATION. All rights reserved.
Copyright 2017 The TensorFlow Authors. All rights reserved.

Various files include modifications (c) NVIDIA CORPORATION. All rights reserved.
NVIDIA modifications are covered by the license terms that apply to the underlying project or file.
```

## 6. Use one of the following methods to test TensorFlow:

- o Test TensorFlow in a simple manner.



```
python
```

```
import tensorflow as tf
hello = tf.constant('Hello, TensorFlow!')
sess = tf.Session()
sess.run(hello)
```

If TensorFlow loads the GPU as expected, the returned information is similar to the command output in the following figure.

```
root@... # python
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
>>> hello = tf.constant('Hello, TensorFlow!')
>>> sess = tf.Session()
2018-03-30 03:37:53.682157: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:892] s
be at least one NUMA node, so returning NUMA node zero
2018-03-30 03:37:53.682544: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1030] Foun
name: Tesla P100-PCIE-16GB major: 6 minor: 0 memoryClockRate(GHz): 1.3285
pciBusID: 0000:00:08:0
totalMemory: 15.89GiB freeMemory: 15.60GiB
2018-03-30 03:37:53.682583: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1120] Crea
16GB, pci bus id: 0000:00:08:0, compute capability: 6.0)
>>> sess.run(hello)
b'Hello, TensorFlow!'
>>>
```

- o Download the TensorFlow model and test TensorFlow.

```
git clone https://github.com/tensorflow/models.git
cd models/tutorials/image/alexnet
python alexnet_benchmark.py --batch_size 128 --num_batches 100
```

The following figure shows the running status of TensorFlow.

```
conv1 [128, 56, 56, 64]
pool1 [128, 27, 27, 64]
conv2 [128, 27, 27, 192]
pool2 [128, 13, 13, 192]
conv3 [128, 13, 13, 384]
conv4 [128, 13, 13, 256]
conv5 [128, 13, 13, 256]
pool5 [128, 6, 6, 256]
2018-03-30 03:40:13.357785: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:892] successful NUMA node read from SysFS
be at least one NUMA node, so returning NUMA node zero
2018-03-30 03:40:13.358207: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1030] Found device 0 with properties:
name: Tesla P100-PCIE-16GB major: 6 minor: 0 memoryClockRate(GHz): 1.3285
pciBusID: 0000:00:08:0
totalMemory: 15.89GiB freeMemory: 15.60GiB
2018-03-30 03:40:13.358245: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1120] Creating TensorFlow device (/device:GPU:
16GB, pci bus id: 0000:00:08:0, compute capability: 6.0)
2018-03-30 03:40:15.916471: step 0, duration = 0.038
2018-03-30 03:40:16.299169: step 10, duration = 0.038
2018-03-30 03:40:16.682881: step 20, duration = 0.038
2018-03-30 03:40:17.065379: step 30, duration = 0.038
2018-03-30 03:40:17.448118: step 40, duration = 0.038
2018-03-30 03:40:17.830372: step 50, duration = 0.038
2018-03-30 03:40:18.213018: step 60, duration = 0.038
2018-03-30 03:40:18.595734: step 70, duration = 0.038
2018-03-30 03:40:18.978311: step 80, duration = 0.038
2018-03-30 03:40:19.361063: step 90, duration = 0.038
2018-03-30 03:40:19.705396: Forward across 100 steps, 0.038 +/- 0.000 sec / batch
2018-03-30 03:40:21.164735: step 0, duration = 0.090
2018-03-30 03:40:22.062778: step 10, duration = 0.090
2018-03-30 03:40:22.962202: step 20, duration = 0.090
2018-03-30 03:40:23.860856: step 30, duration = 0.090
2018-03-30 03:40:24.758891: step 40, duration = 0.090
2018-03-30 03:40:25.657170: step 50, duration = 0.090
2018-03-30 03:40:26.555194: step 60, duration = 0.090
2018-03-30 03:40:27.452843: step 70, duration = 0.090
2018-03-30 03:40:28.351892: step 80, duration = 0.090
2018-03-30 03:40:29.249606: step 90, duration = 0.090
2018-03-30 03:40:30.058889: Forward-backward across 100 steps, 0.090 +/- 0.000 sec / batch
```

7. Save the settings that you configured for the TensorFlow image. If you do not save the settings that you configured for the TensorFlow image, the settings cannot take effect the next time you log on to the instance.

# 3. Use RAPIDS to accelerate machine learning tasks on a GPU-accelerated instance

This topic describes how to use the NGC-based Real-time Acceleration Platform for Integrated Data Science (RAPIDS) libraries that are installed on a GPU-accelerated instance to accelerate tasks for data science and machine learning as well as improve the efficiency of computing resources.

## Background information

RAPIDS is an open source suite of data processing and machine learning libraries developed by NVIDIA to enable GPU-acceleration for data science and machine learning. For more information about RAPIDS, visit the [RAPIDS website](#).

NVIDIA GPU Cloud (NGC) is a deep learning ecosystem developed by NVIDIA to provide developers with free access to deep learning and machine learning software stacks to build corresponding development environments. The [NGC website](#) provides RAPIDS Docker images, which come pre-installed with development environments.

JupyterLab is an interactive development environment that makes it easy to browse, edit, and run code files on your servers.

Dask is a lightweight big data framework that can make parallel computing more efficient.

This topic provides sample code that is modified based on the NVIDIA RAPIDS Demo code and datasets and demonstrates how to use RAPIDS to accelerate an end-to-end task from the Extract-Transform-Load (ETL) phase to the Machine Learning (ML) Training phase on a GPU-accelerated instance. The RAPIDS cuDF library is used in the ETL phase and the XGBoost model is used in the ML Training phase. The sample code is based on the Dask framework and runs on a single machine.

To obtain the official RAPIDS Demo code of NVIDIA, visit [Mortgage Demo](#).

## Procure

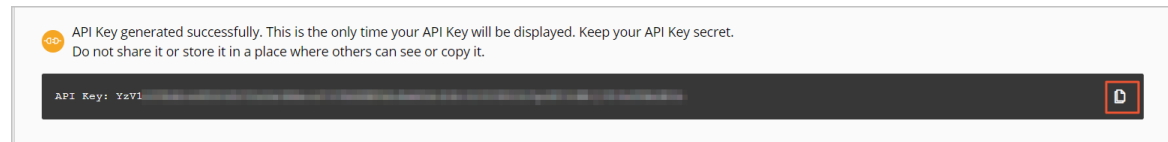
- [Step 1: Obtain the NGC API key](#)
- [Step 2: Obtain the RAPIDS image download command](#)
- [Step 3: Deploy the RAPIDS environment](#)
- [Step 4: Run RAPIDS Demo](#)

## Step 1: Obtain the NGC API key

1. Create an account on the [NGC registration page](#).
2. Log on to the [NGC website](#).
3. In the upper-right corner, click the username and select **Get API Key**.
4. Click **Generate API Key**.
5. On the **Generate a New API Key**, select **Confirm**.

A new NGC API key overwrites the previous API key. Before you obtain a new API key, you must make sure that the previous API key is no longer needed.

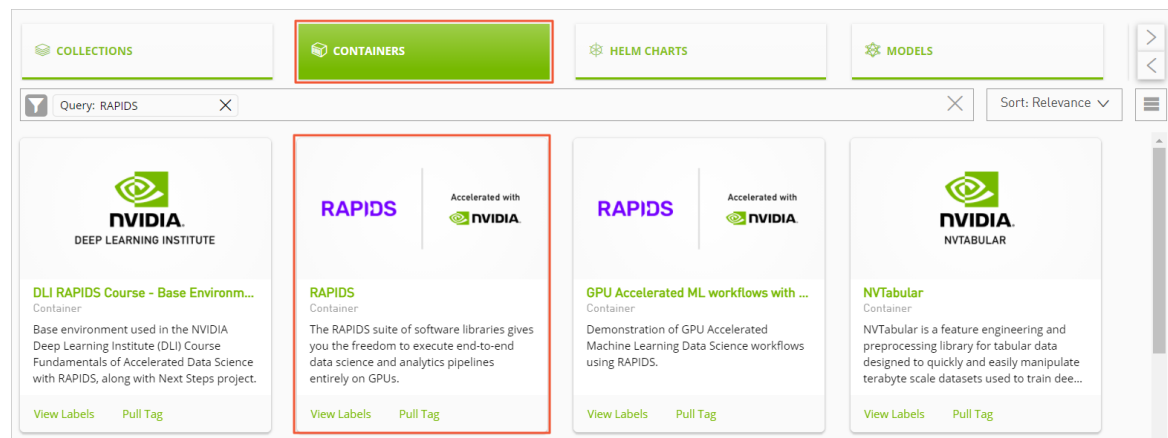
6. Copy the API key and save it to your local storage device.



## Step 2: Obtain the RAPIDS image download command

Perform the following steps to obtain the download command of the RAPIDS image:

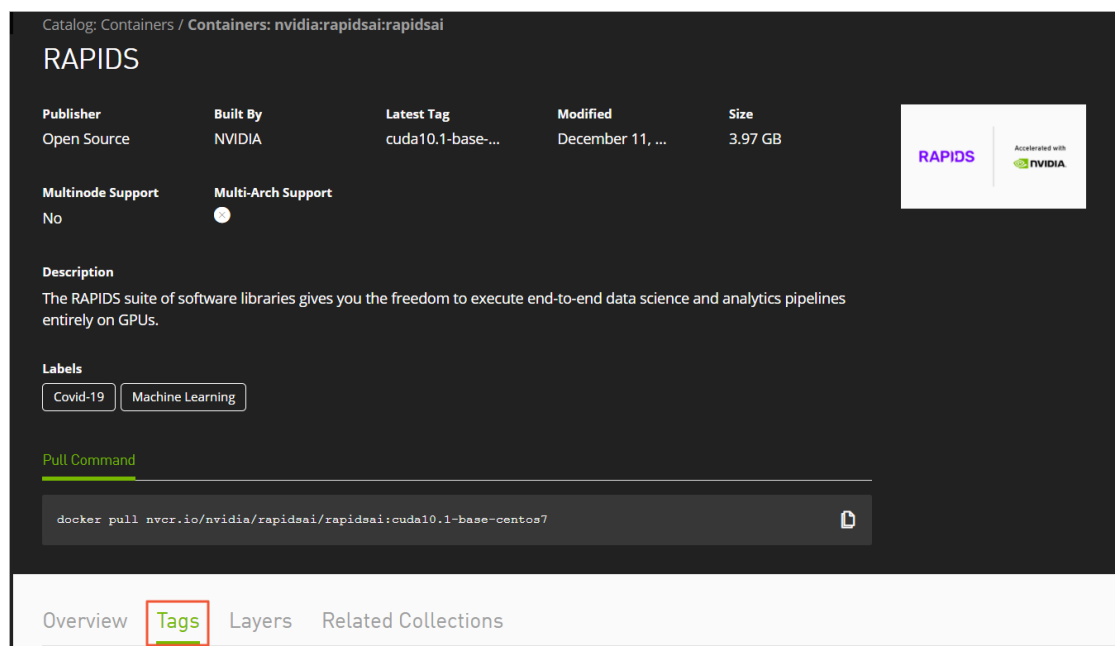
1. Log on to the [NGC website](#).
2. On the page that appears, click the **RAPIDS** image.



3. Obtain the docker pull command.

The sample code in this topic is based on the RAPIDS v0.8 image. Therefore, use the image that is tagged with v0.8 when you run the sample code. If you use another image, the corresponding commands may differ.

- i. On the RAPIDS page, click the **Tags** tab.



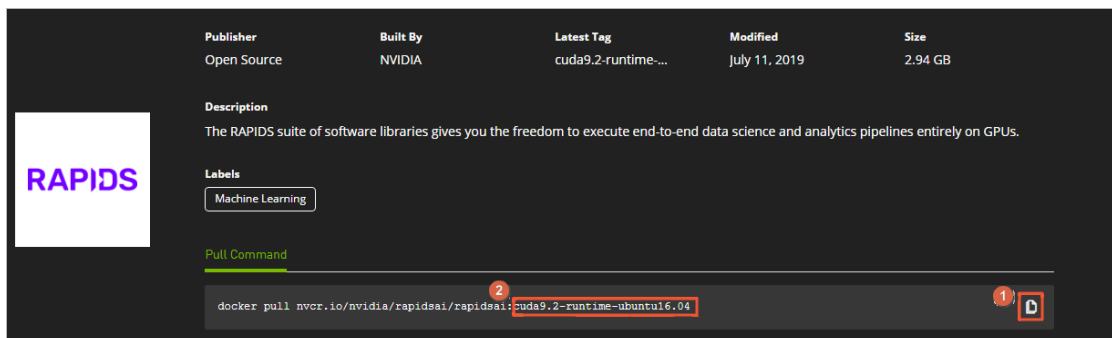
- ii. Copy the tag information. `0.8-cuda10.0-runtime-ubuntu16.04-gcc5-py3.6` is copied in this example.



- iii. Return to the top of the page, find the **Pull Command** section, and copy the displayed command. Paste the copied command to the text editor. Then, replace the image version with the tag information obtained in the preceding step and save the file.

In this example, `cuda9.2-runtime-ubuntu16.04` is replaced with `0.8-cuda10.0-runtime-ubuntu16.04-gcc5-py3.6`.

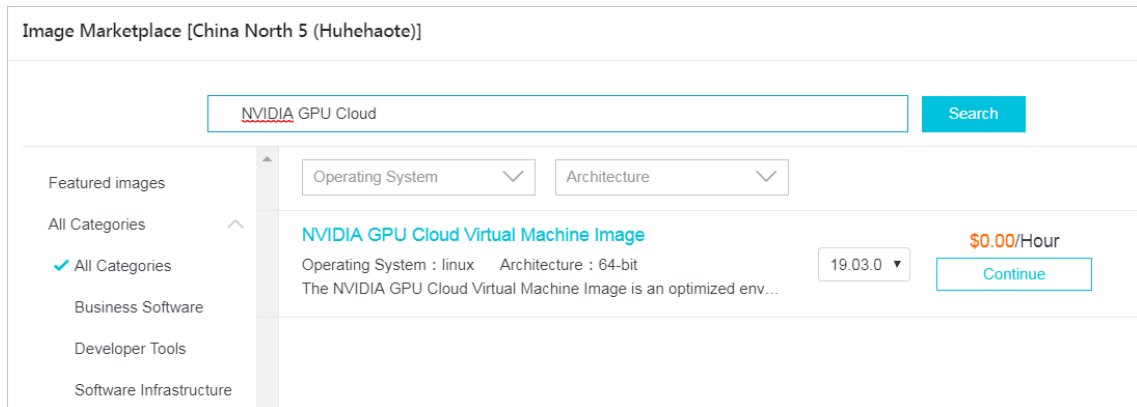
The saved docker pull command is used to download the RAPIDS image. For more information about how to download the RAPIDS image, see the [Step 3: Deploy the RAPIDS environment](#) section.



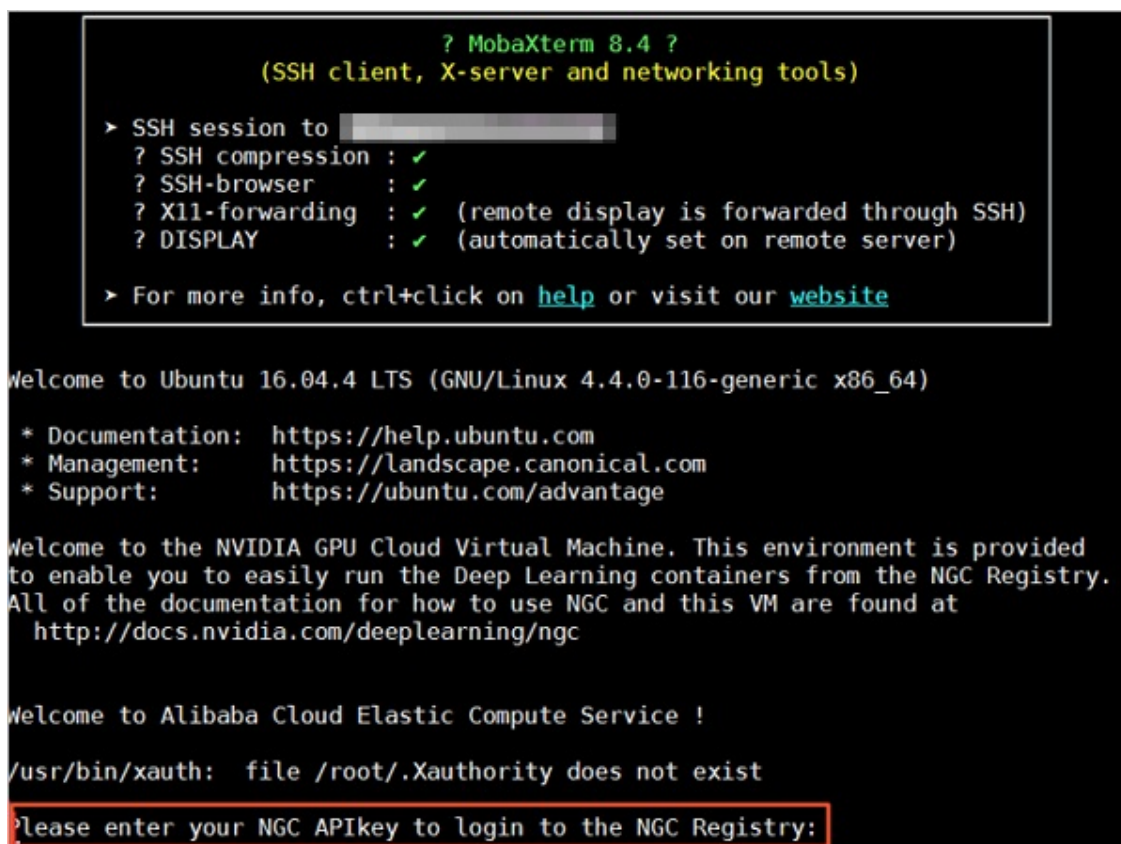
## Step 3: Deploy the RAPIDS environment

Perform the following steps to deploy the RAPIDS environment:

1. Create a GPU-accelerated instance. For more information, see [Create an instance by using the wizard](#).  
Configure the following parameters:
  - Instance Type: RAPIDS applies only to GPU models that use the NVIDIA Pascal or a later architecture. Therefore, you must select an instance type that meets the GPU requirements. The following instance families are available: gn6i, gn6v, gn5, and gn5i. For more information, see [Instance family](#). We recommend that you select an instance family that has larger video memory, such as gn6i, gn6v, or gn5. The GPU-accelerated instance that has a 16 GB video memory is used in this example.
  - Image: Select `NVIDIA GPU Cloud Virtual Machine Image` in the Image Marketplace dialog box.



- Public IP Address: Select **Assign Public IPv4 Address** or attach an elastic IP address (EIP) after you create the GPU-accelerated instance. For more information, see [Associate an EIP with an ECS instance](#).
  - Security Group: Select a security group for which the following ports are enabled:
    - TCP port 22, used for SSH logon
    - TCP port 8888, used to access JupyterLab
    - TCP port 8786 and TCP port 8787, used to access Dask
2. Connect to the GPU-accelerated instance. For more information, see [Connection methodsGuidelines on instance connection](#).
  3. Enter the NGC API key and press the Enter key to log on to the NGC container.



4. Run the `nvidia-smi` command to view GPU information, such as the GPU model and GPU driver version.

We recommend that you check the GPU information to identify potential issues. For example, if an earlier NGC driver version is used, it may not be supported by the Docker image.

5. Run the docker pull command to download the RAPIDS image.

For more information about how to obtain the docker pull command, see the [Step 2: Obtain the RAPIDS image download command](#) section.

```
docker pull nvcr.io/nvidia/rapidsai/rapidsai:0.8-cuda10.0-runtime-ubuntu16.04-gcc5-py3.6
```

6. **Optional:** View the downloaded image.

We recommend that you view the Docker image information to ensure that the correct image is downloaded.

```
docker images
```

7. Run the NGC container to deploy the RAPIDS environment.

```
docker run --runtime=nvidia \
  --rm -it \
  -p 8888:8888 \
  -p 8787:8787 \
  -p 8786:8786 \
  nvcr.io/nvidia/rapidsai/rapidsai:0.8-cuda10.0-runtime-ubuntu16.04-gcc5-py3.6
```

## Step 4: Run RAPIDS Demo

Perform the following steps to run RAPIDS Demo:

1. Download the dataset and the Demo file on the GPU-accelerated instance.

```
# Get apt source address and download demos.
source_address=$(curl http://100.100.100.200/latest/meta-data/source-address|head -n 1)
source_address="${source_address}/opsx/ecs/linux/binary/machine_learning/"
cd /rapids
wget $source_address/rapids_notebooks_v0.8.tar.gz
tar -xzf rapids_notebooks_v0.8.tar.gz
cd /rapids/rapids_notebooks_v0.8/xgboost
wget $source_address/data/mortgage/mortgage_2000_1gb.tgz
```

2. Start JupyterLab on the GPU-accelerated instance.

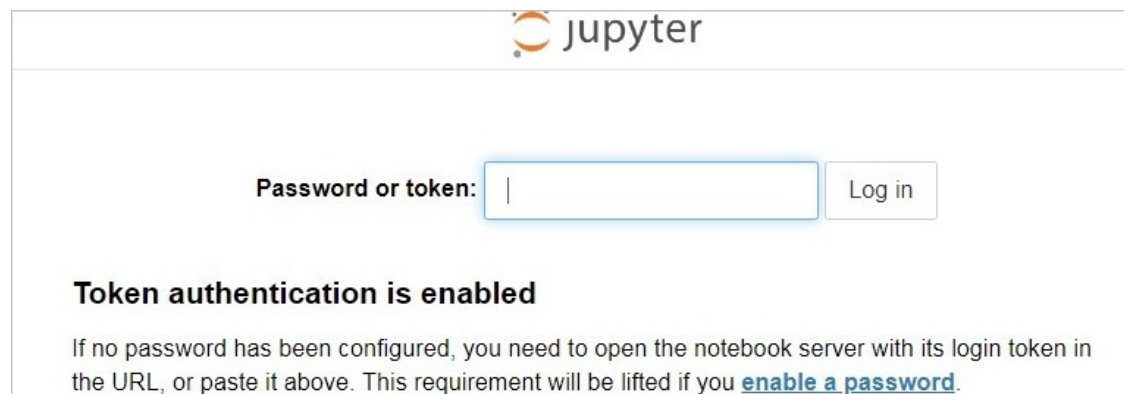
We recommend that you run the commands to start JupyterLab directly.

```
# Run the following command to start JupyterLab and set the password.
cd /rapids/rapids_notebooks_v0.8/xgboost
jupyter-lab --allow-root --ip=0.0.0.0 --no-browser --NotebookApp.token='YOUR_PASSWORD'
# Exit JupyterLab.
sh ../utils/stop-jupyter.sh
```

- o You can also run the `sh ../utils/start-jupyter.sh` script to start JupyterLab. However, you cannot set the login password if you run the script.
  - o You can also press `Ctrl+C` twice to exit JupyterLab.
3. Open your browser and enter `http://IP address of your GPU-accelerated instance:8888` in the address bar to access JupyterLab.



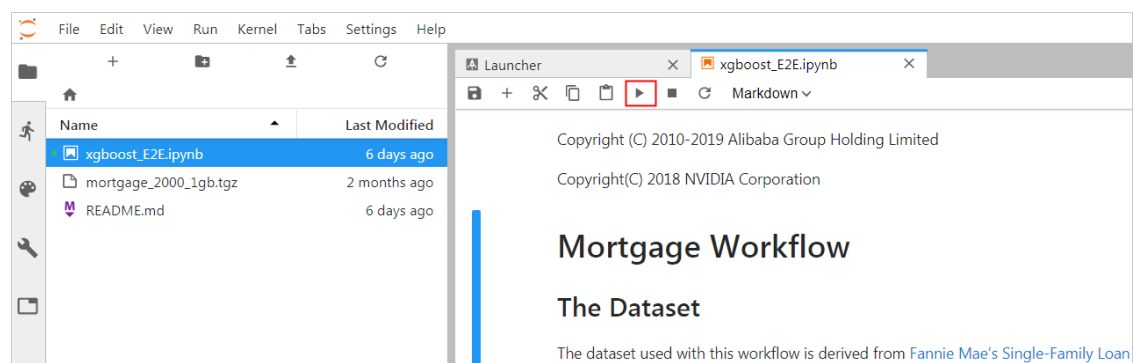
If you set the logon password when you start JupyterLab, you are directed to a page to enter your password.



#### 4. Run the Notebook code.

A mortgage repayment task is used in this example. For more information, see the [Code running](#) section. The Notebook code includes the following content:

- *xgboost\_E2E.ipynb*: an XGBoost Demo file. You can double-click this file to view its details, or click the Execute icon to run one cell at a time.



- *mortgage\_2000\_1gb.tgz*: a file named mortgage\_2000\_1gb.tgz, which contains the mortgage repayment training data of year 2000. Files in the perf folder are split into files of 1 GB to maximize the usage of GPU video memory.

## Code running

In this example, XGBoost is used to demonstrate the end-to-end code running process from data pre-processing to training the XGBoost data model. The process involves the following phases:

- ETL: performed on the GPU-accelerated instance in most cases. Data is extracted, transformed, and then loaded onto the data warehouse.
- Data Conversion: performed on the GPU-accelerated instance. Data processed in the ETL phase is converted into the DMatrix format so that it can be used by XGBoost to train the data model.
- ML-Training: performed on the GPU-accelerated instance by default. XGBoost is used to train the gradient boosting decision tree (GBDT).

Perform the following steps to run the Notebook code:

#### 1. Prepare the dataset.

In this example, the shell script downloads the mortgage repayment training data of year 2000 by default, which is the mortgage\_2000\_1gb.tgz file.

If you want to obtain more data for XGBoost model training, you can set the `download_url` parameter to specify the URL. For more information, visit [Mortgage Data](#).

The following figure shows an example.

```
[ ]: # Open: https://docs.rapids.ai/datasets/mortgage-data in your browser. On the displayed page, two datasets are provided: "Dataset" and "1GB Splits",
# 1GB splits are the same data with the individual performance data files split into 1GB pieces.
# This is useful for GPUs with less memory. We recommend that you download the "1GB Splits" dataset for this example.
# You can download and decompress datasets by setting the "download_url" parameter to the actual download URL.
# eg: download_url = 'http://rapidsai-data.s3-website.us-east-2.amazonaws.com/notebook-mortgage-data/mortgage_2000-xxxxx_1gb.tgz'
download_url = '' # if download_url = '', use downloaded dataset(mortgage_2000-2001_1gb).

if download_url != '':
    # get download filename from url
    download_filename = download_url.split('/')[-1]
    # the directory to which the file is decompressed.
    mortgage_dir = download_filename.split('.')[0]
    # download dataset with url and filename, if the current directory exist filename, the filename will not be downloaded again.
    download_file_from_url(download_url, download_filename)
    # decompress xxx.tgz to mortgage_dir. if the folder mortgage_dir already exist, the filename will not be decompressed again.
    decompress_file(download_filename, mortgage_dir)
else:
    # use default downloaded data
    mortgage_dir = 'mortgage_2000_1gb'
    # decompress xxx.tgz to mortgage_dir. if the folder mortgage_dir already exist, the filename will not be decompressed again.
    decompress_file('mortgage_2000_1gb.tgz', mortgage_dir)
```

## 2. Set relevant parameters.

Parameter	Description
<b>start_year</b>	Specifies the start year from which the training data is selected. In the ETL phase, data generated between the <b>end_year</b> range is processed.
<b>end_year</b>	Specifies the end year from which the training data is selected. In the ETL phase, data generated between the <b>end_year</b> range is processed.
<b>train_with_gpu</b>	Specifies whether to use GPU for XGBoost model training. Default value: True.
<b>gpu_count</b>	Specifies the number of workers to be started. Default value: 1. You can set the parameter based on your actual needs but the value must be less than the number of GPUs in the GPU-accelerated instance.
<b>part_count</b>	Specifies the number of performance files used for data model training. Default value: 2 × <b>gpu_count</b> . If the value is too large, an insufficient memory error occurs in the Data Conversion phase and the error message is stored in the background of Notebook.

The following figure shows an example.



```
[5]: acq_data_path = "{}acq".format(mortgage_dir)
    perf_data_path = "{}perf".format(mortgage_dir)
    col_names_path = "{}names.csv".format(mortgage_dir)

    start_year = 2000
    end_year = 2000 # the end_year is inclusive

    # whether use GPUs for XGBoost training
    train_with_gpu = True

    # The number of GPUs to be used, value range: 1 to get_gpu_nums(). Default value: 1.
    # This parameter would use for starting dask-worker, doing ETL, doing Conversion and training model(if train_with_gpu=True).
    gpu_count = 1

    # The number of performance files in the perf folder
    part_number = len(os.listdir(perf_data_path))

    # if you download 1GB Splits train data(the filename end with '1gb.tgz'), each performance file is no large than 1GB,
    # in this example, a 16G GPU can process 2 or 3 performance files. By default, one GPU is set to process 2 files.
    part_count = 2 * gpu_count if part_number >= 2 * gpu_count else part_number

    print('>>> Using "{}" GPU(GPUs)'.format(gpu_count))
    print('>>> ETL - process performance files from "{}" to "{}"'.format(start_year, end_year))
    print('>>> Data Conversion - select "{}" performance data processed in the ETL phase and convert to DMatrix-format for XGBoost training.'.format(part_count))
    print('>>> ML - Whether to use the GPU for XGBoost training: "{}"'.format(train_with_gpu))

    >>> Using "1" GPU(GPUs).
    >>> ETL - process performance files from "2000" to "2000".
    >>> Data Conversion - select "2" performance data processed in the ETL phase and convert to DMatrix-format for XGBoost training.
    >>> ML - Whether to use the GPU for XGBoost training: "True".
```

### 3. Start Dask.

The Notebook code starts Dask Scheduler, and also starts workers based on the `gpu_count` value for ETL and data model training. After you start Dask, you can monitor tasks on the Dask Dashboard. For more information about how to start Dask, see the [Dask Dashboard](#) section.

The following figure shows an example.

```
# run dask-worker
cmd = "hostname --all-ip-addresses"
process = subprocess.Popen(cmd.split(), stdout=subprocess.PIPE)
output, error = process.communicate()
IPADDR = str(output.decode()).split()[0]

cluster = LocalCUDACluster(n_workers=gpu_count, ip=IPADDR)
client = Client(cluster)
client
```

Client	Cluster
<ul style="list-style-type: none"> <li>Scheduler: tcp://172.17.0.2:43894</li> <li>Dashboard: http://172.17.0.2:8787/status</li> </ul>	<ul style="list-style-type: none"> <li>Workers: 1</li> <li>Cores: 1</li> <li>Memory: 507.25 GB</li> </ul>

### 4. Start the ETL phase.

In this phase, tables are joined, grouped, aggregated, and sliced. The data format is DataFrame of the cuDF library, which is similar to Pandas DataFrame.

The following figure shows an example.

## ETL

Perform all of ETL with a single call to

```
process_quarter_gpu(year=year, quarter=quarter, perf_file=file)
```

```
]: %%time

# NOTE: The ETL calculates additional features which are then dropped before creating the XGBoost DMatrix.
# This can be optimized to avoid calculating the dropped features.

gpu_dfs = []
gpu_time = 0
quarter = 1
year = start_year
count = 0
while year <= end_year:
    for file in glob(os.path.join(perf_data_path + "/Performance_" + str(year) + "Q" + str(quarter) + "*")):
        gpu_dfs.append(process_quarter_gpu(year=year, quarter=quarter, perf_file=file))
        count += 1
    quarter += 1
    if quarter == 5:
        year += 1
        quarter = 1
wait(gpu_dfs)

CPU times: user 560 ms, sys: 28 ms, total: 588 ms
Wall time: 20.9 s
```

### 5. Start the Data Conversion phase.

In this phase, DataFrame-format data is converted into the DMatrix-format data for XGBoost model training. Each worker processes one DMatrix object.

The following figure shows an example.

Load the data from host memory, and convert to CSR

```
%%time

gpu_dfs = [delayed(DataFrame.from_arrow)(gpu_df) for gpu_df in gpu_dfs[:part_count]]
gpu_dfs = [gpu_df for gpu_df in gpu_dfs]
wait(gpu_dfs)

tmp_map = [(gpu_df, list(client.who_has(gpu_df).values())[0]) for gpu_df in gpu_dfs]
new_map = {}
for key, value in tmp_map:
    if value not in new_map:
        new_map[value] = [key]
    else:
        new_map[value].append(key)

del(tmp_map)
gpu_dfs = []
for list_delayed in new_map.values():
    gpu_dfs.append(delayed(cudf.concat)(list_delayed))

del(new_map)
gpu_dfs = [(gpu_df[['delinquency_12']], gpu_df[delayed(list)(gpu_df.columns.difference(['delinquency_12'])))]) for gpu_df in gpu_dfs]
gpu_dfs = [(gpu_df[0].persist(), gpu_df[1].persist()) for gpu_df in gpu_dfs]

gpu_dfs = [dask.delayed(xgb.DMatrix)(gpu_df[1], gpu_df[0]) for gpu_df in gpu_dfs]
gpu_dfs = [gpu_df.persist() for gpu_df in gpu_dfs]
gc.collect()
wait(gpu_dfs)

CPU times: user 200 ms, sys: 4 ms, total: 204 ms
Wall time: 4.3 s
```

### 6. Start the ML Training phase.

In this phase, data model training is started by dask-xgboost, which supports collaborative communication among Dask workers. At the bottom layer, dask-xgboost is also called to execute data model training.

The following figure shows an example.

```

Train the Gradient Boosted Decision Tree with a single call to

dask_xgboost.train(client, params, data, labels, num_boost_round=dxgb_gpu_params['nround'])

[ ]: if train_with_gpu:
    print('>>> Training with {} GPU.'.format(gpu_count))
else:
    print('>>> Training with {} CPU.'.format(gpu_count))
    dxgb_gpu_params['tree_method'] = 'hist'

print('>>> Worker number: {}'.format(gpu_count))
print('>>> part_count: {}'.format(part_count))

rows = sum([dmatrix.num_row().compute() for dmatrix in gpu_dfs])
cols = gpu_dfs[0].num_col().compute()
print('>>> Train data rows: {},\tcols: {}'.format(rows, cols))

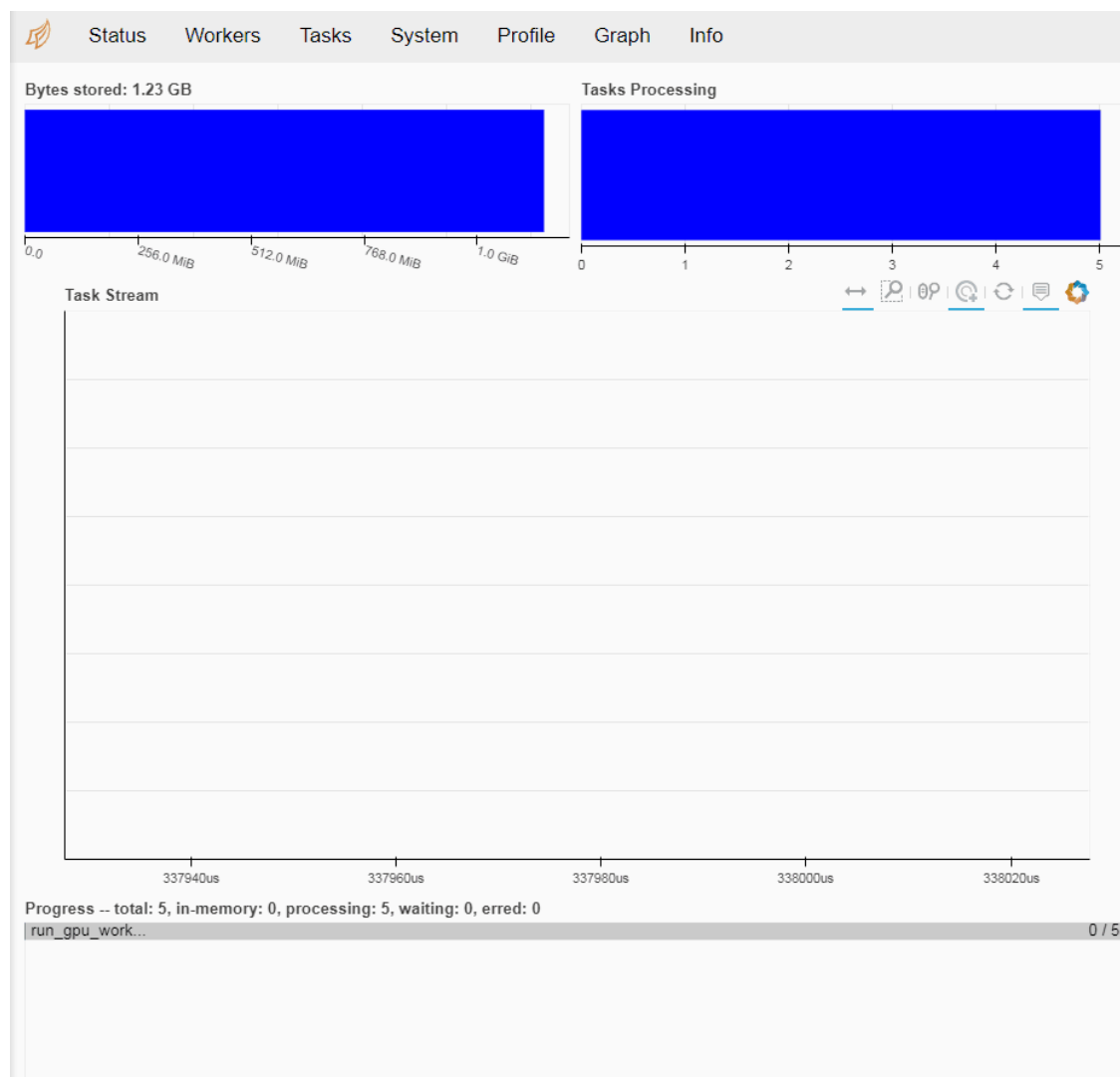
[ ]: %%time
labels = None
bst = dxgb_gpu.train(client, dxgb_gpu_params, gpu_dfs, labels, num_boost_round=dxgb_gpu_params['nround'])

```

## Dask Dashboard

Dask Dashboard supports task progress tracking, task performance problem identification, and fault debugging.

After Dask is started, you can enter `http://IP address of your GPU-accelerated instance:8787/status` in the address bar of your browser to go to the Dask Dashboard page.



## Related functions

Operation	Function name
Download a file.	<code>def download_file_from_url(url, filename):</code>
Decompress a file.	<code>def decompress_file(filename, path):</code>
Obtain the number of GPUs in the current machine.	<code>def get_gpu_nums():</code>
Manage the GPU memory.	<ul style="list-style-type: none"> <li>• <code>def initialize_rmm_pool():</code></li> <li>• <code>def initialize_rmm_no_pool():</code></li> <li>• <code>def run_dask_task(func, **kwargs):</code></li> </ul>
Submit a Dask task.	<ul style="list-style-type: none"> <li>• <code>def process_quarter_gpu(year=2000, quarter=1, perf_file=""):</code></li> <li>• <code>def run_gpu_workflow(quarter=1, year=2000, perf_file="", **kwargs):</code></li> </ul>
Use cuDF to load data from a CSV file.	<ul style="list-style-type: none"> <li>• <code>def gpu_load_performance_csv(performance_path, **kwargs):</code></li> <li>• <code>def gpu_load_acquisition_csv(acquisition_path, **kwargs):</code></li> <li>• <code>def gpu_load_names(**kwargs):</code></li> </ul>
Process and extract characteristics of data for training machine learning models.	<ul style="list-style-type: none"> <li>• <code>def null_workaround(df, **kwargs):</code></li> <li>• <code>def create_ever_features(gdf, **kwargs):</code></li> <li>• <code>def join_ever_delinq_features(everdf_tmp, delinq_merge, **kwargs):</code></li> <li>• <code>def create_joined_df(gdf, everdf, **kwargs):</code></li> <li>• <code>def create_12_mon_features(joined_df, **kwargs):</code></li> <li>• <code>def combine_joined_12_mon(joined_df, testdf, **kwargs):</code></li> <li>• <code>def final_performance_delinquency(gdf, joined_df, **kwargs):</code></li> <li>• <code>def join_perf_acq_gdfs(perf, acq, **kwargs):</code></li> <li>• <code>def last_mile_cleaning(df, **kwargs):</code></li> </ul>