Contents

3.11.2 Preparing the Hardware Environment ..................................................................................................................... 27
3.12 A800-9010 .................................................................................................................................................................................. 28
3.12.1 Installation Scheme .................................................................................................................................................................. 28
3.12.2 Preparing the Hardware Environment ................................................................................................................................. 30
3.13 A900-9000 ..................................................................................................................................................................................... 31
3.13.1 Installation Scheme .................................................................................................................................................................. 31
3.13.2 Preparing the Hardware Environment ................................................................................................................................. 33

4 Installing the Driver and Firmware .......................................................................................................................................... 34

5 Installing the Development Environment ........................................................................................................................................ 36
5.1 Obtaining Software Packages ..................................................................................................................................................... 36
5.2 Preparing Installation and Running Users ................................................................................................................................. 38
5.3 Installing OS Dependencies ......................................................................................................................................................... 39
5.3.1 Before You Start ....................................................................................................................................................................... 39
5.3.2 Dependency List ............................................................................................................................................................................ 39
5.3.3 Installation Procedure (Ubuntu 18.04) ........................................................................................................................................ 42
5.3.4 Installation Procedure (CentOS 7.6) ........................................................................................................................................... 46
5.4 Installing Software Packages on a Non-Ascend Device .................................................................................................................................................. 50
5.4.1 Installing the Development Kit .............................................................................................................................................. 50
5.4.2 Configuring the Cross Compilation Environment ....................................................................................................................... 51
5.4.3 Follow-up .......................................................................................................................................................................................... 52
5.5 Installing Software Packages on an Ascend Device ....................................................................................................................... 52
5.5.1 Installing the Development Kit .............................................................................................................................................. 52
5.5.2 Installing the Framework Plugin Package ................................................................................................................................. 53
5.5.3 Installing the Deep Learning Framework ................................................................................................................................... 54
5.5.4 Installing protobuf Python ......................................................................................................................................................... 55
5.5.5 Configuring the NIC IP Address of a Device ............................................................................................................................... 57
5.5.6 Configuring Environment Variables ......................................................................................................................................... 59
5.5.7 Follow-up Procedure ...................................................................................................................................................................... 59

6 Installing the Operating Environment (Deploying the nnrt Software) .......................................................................................... 60
6.1 Before You Start .............................................................................................................................................................................. 60
6.2 Obtaining Software Packages ....................................................................................................................................................... 61
6.3 Installing the Operating Environment on the PM or VM ............................................................................................................. 62
6.3.1 Preparing Installation and Running Users ............................................................................................................................... 62
6.3.2 Installing the Offline Inference Engine Package ....................................................................................................................... 63
6.3.3 Configuring Environment Variables ....................................................................................................................................... 64
6.3.4 Follow-up .......................................................................................................................................................................................... 64

7 Installing the Operating Environment (Deploying the nnae Software) .......................................................................................... 65
7.1 Before You Start .............................................................................................................................................................................. 65
7.2 Obtaining Software Packages ....................................................................................................................................................... 66
7.3 Installing the Operating Environment on the PM or VM ............................................................................................................. 67
7.3.1 Preparing Installation and Running Users ............................................................................................................................... 67
1 Installation Precautions

Huawei Compute Architecture for Neural Networks (CANN) is a heterogeneous computing architecture for AI scenarios and provides multi-layer programming interfaces to help users quickly build AI applications and services based on the Ascend platform.

- Development environment: used for development activities such as coding, compilation, and debugging.
  - (Scenario 1) The development environment installed on an Ascend AI device can also be used as an operating environment to run applications or port, develop, and debug training scripts.
  - (Scenario 2) The development environment installed on a non-Ascend AI device can only be used for development activities that do not depend on Ascend devices, such as compilation, model conversion using the ATC, and coding for operators and inference applications.
- Operating environment: used for running user applications or port, develop, and debug training scripts on Ascend AI devices.
Figure 2-1 shows the process of installing the development or operating environment.

Figure 2-1 Installation flowchart
3 Preparations for Installation

3.1 Non-Ascend Devices

3.1.1 Installation Scheme

For non-Ascend AI devices, only the pure development environment can be installed. Figure 3-1 shows the software to be installed.

No firmware or driver needs to be installed on non-Ascend AI devices. Non-Ascend AI devices can be used only for code development and compilation that do not depend on Ascend devices.

Software packages:

Toolkit: development kit. It is used for application development, operator customization, and model conversion. The development kit contains the library
files required for developing applications and auxiliary development tools such as the ATC for model conversion.

For details about the software packages, see 11 Software Packages.

Figure 3-1 Development environment

![Development environment](image)

**NOTE**

If the development environment uses the x86_64 architecture and the operating environment uses the AArch64 architecture, the ACLib library of the AArch64 architecture development kit needs to be called during application compilation because the development kits of both the x86_64 and AArch64 architectures need to be deployed in the development environment.

3.2 A200-3000 (EP Scenario)

3.2.1 Installation Scheme

Atlas 200 AI accelerator modules (model 3000) are distinguished based on the working mode of the PCIe of the Ascend 310 AI Processor. If the PCIe of the Ascend 310 AI Processor works in master mode, peripherals can be extended. This mode is called the RC mode. If the PCIe of the Ascend 310 AI Processor works in slave mode, it is called the EP mode. For details, see 13.4 Product Modes in Inference Scenario.

- Development environment

  Figure 3-2 shows the software to be installed in the development environment.

  Software packages:
  - npu-firmware: firmware installation package
  - npu-driver: driver installation package
  - Toolkit: development kit. It is used for application development, operator customization, and model conversion. A development kit contains the library files required for developing applications and auxiliary development tools such as the Ascend Tensor Compiler (ATC) for model conversion.

  For details about the software packages, see 11 Software Packages.
Operating environment

**Figure 3-3** shows the software to be installed in the operating environment. Software packages:

- npu-firmware: firmware installation package
- npu-driver: driver installation package
- nnrt: offline inference engine package. It contains the ACL library (ACLlib), which is used for model inference for applications. Only offline inference is supported.

For details about the software packages, see [11 Software Packages](#).

**Figure 3-3** Operating environment

---

NOTE

- Offline inference: converts models into OM models based on the original AI framework and does not depend on AI frameworks for inference.
- Online inference: ports applications that use the original AI framework for inference to Ascend AI Processors. In this scenario, AI frameworks are needed for inference.
3.2.2 Preparing the Hardware Environment

Table 3-1 lists the OSs supported by an Atlas 200 AI accelerator module (model 3000) when it is used as a PCIe slave device (EP mode).

<table>
<thead>
<tr>
<th>OS</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
</table>

3.3 A200-3000 (RC Scenario)

3.3.1 Installation Scheme

Atlas 200 AI accelerator modules (model 3000) are distinguished based on the working mode of the PCIe of the Ascend 310 AI Processor. If the PCIe of the Ascend 310 AI Processor works in master mode, peripherals can be extended. This mode is called the RC mode. If the PCIe of the Ascend 310 AI Processor works in slave mode, it is called the EP mode.

A200-3000 (RC scenario) can only be used as the operating environment.

- Development environment
  You need to prepare another server to set up the development environment. For details, see 3.1 Non-Ascend Devices.

- Operating environment
  - The boot image package of the Atlas 200 AI accelerator module needs to be created in the Ubuntu 18.04 environment. See the Atlas 200 AI Accelerator Module 1.0.10 Software Installation and Maintenance Guide (RC, Model 3000) for the operating environment installation.

Figure 3-4 shows the installation diagram.

For details about the software packages, see 11 Software Packages.
3.4 A300I-3000

3.4.1 Installation Scheme

The installation for the Atlas 300I inference card (model 3000) development or operating environment is as follows:

- Development environment

  Figure 3-5 shows the software to be installed in the development environment.

  Software packages:
  - npu-firmware: firmware installation package
  - npu-driver: driver installation package
  - Toolkit: development kit. It is used for application development, operator customization, and model conversion. A development kit contains the library files required for developing applications and auxiliary development tools such as the Ascend Tensor Compiler (ATC) for model conversion.
  - tfplugin: plugin package, an adaptation plugin used to connect to the upper-layer framework TensorFlow. If the deep learning framework TensorFlow is used for online inference, you need to install the software package.
  - AI framework: Only TensorFlow and MindSpore frameworks are supported. Before performing online inference, install an AI framework.

  For details about the software packages, see 11 Software Packages.
Operating environment

Figure 3-6 shows the software to be installed in the operating environment.

Software packages:
- npu-firmware: firmware installation package
- npu-driver: driver installation package
- nnrt: offline inference engine package. It contains the ACL library (ACLlib), which is used for model inference of applications. nnrt supports offline inference only. This software package is installed in the Atlas Intelligent Edge Solution scenario.
- nnae: deep learning engine package. It contains the FWK library (Fwklib) and OPP. nnae supports offline and online inference (with TensorFlow and MindSpore frameworks only). This software package is installed in the Ascend Data Center Solution scenario.
- tfplugin: plugin package, an adaptation plugin used to connect to the upper-layer framework TensorFlow. If the deep learning framework TensorFlow is used for online inference, you need to install the software package.
- AI framework: Only TensorFlow and MindSpore frameworks are supported. Before performing online inference, install an AI framework.

For details about the software packages, see 11 Software Packages.
3.4.2 Preparing the Hardware Environment

Prepare the hardware environment by referring to the *Atlas 300i Inference Card User Guide (Model 3000)*.

Table 3-2 lists the OSs supported by an Atlas 300i inference card (model 3000).

<table>
<thead>
<tr>
<th>OS</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ubuntu</td>
<td>OS version: Ubuntu 18.04.1</td>
<td>Official Ubuntu website</td>
</tr>
<tr>
<td>CentOS</td>
<td>OS version: CentOS 7.6</td>
<td>Official CentOS website</td>
</tr>
<tr>
<td></td>
<td>Processor architecture: AArch64</td>
<td>Download the recommended version <code>CentOS-7-aarch64-Everything-1810.iso</code> from <a href="https://archive.kernel.org/centos-vault/altarch/7.6.1810/isos/aarch64/">https://archive.kernel.org/centos-vault/altarch/7.6.1810/isos/aarch64/</a>.</td>
</tr>
<tr>
<td>CentOS</td>
<td>OS version: CentOS 8.2</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Processor architecture: AArch64</td>
<td></td>
</tr>
<tr>
<td>EulerOS</td>
<td>OS version: EulerOS 2.8</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Processor architecture: AArch64</td>
<td></td>
</tr>
<tr>
<td>OS</td>
<td>Version</td>
<td>How to Obtain</td>
</tr>
<tr>
<td>--------</td>
<td>--------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>EulerOS</td>
<td>OS version: EulerOS 2.9 Processor architecture: AArch64</td>
<td>-</td>
</tr>
<tr>
<td>Kylin</td>
<td>OS version: Kylin V10 Processor architecture: AArch64</td>
<td>-</td>
</tr>
<tr>
<td>Tlinux</td>
<td>OS version: Tlinux 2.4 Processor architecture: AArch64</td>
<td>-</td>
</tr>
<tr>
<td>NeoKylin</td>
<td>OS version: NeoKylin 7.6 Processor architecture: AArch64</td>
<td>-</td>
</tr>
<tr>
<td>UOS</td>
<td>OS version: UOS V20 SP1 Processor architecture: AArch64</td>
<td>-</td>
</tr>
<tr>
<td>Linx</td>
<td>OS version: Linx 6.0 Processor architecture: AArch64</td>
<td>-</td>
</tr>
</tbody>
</table>

### 3.5 A300I-3010

#### 3.5.1 Installation Scheme

The installation for the Atlas 300I inference card (model 3010) development or operating environment is as follows:

- Development environment

  **Figure 3-7** shows the software to be installed in the development environment.

  Software packages:
  - npu-firmware: firmware installation package
  - npu-driver: driver installation package
  - Toolkit: development kit. It is used for application development, operator customization, and model conversion. A development kit contains the library files required for developing applications and auxiliary development tools such as the Ascend Tensor Compiler (ATC) for model conversion.
  - tfplugin: plugin package, an adaptation plugin used to connect to the upper-layer framework TensorFlow. If the deep learning framework TensorFlow is used for online inference, you need to install the software package.
AI framework: Only TensorFlow and MindSpore frameworks are supported. Before performing online inference, install an AI framework.

For details about the software packages, see **11 Software Packages**.

**Figure 3-7** Development environment

- Operating environment

  **Figure 3-8** shows the software to be installed in the operating environment.

  Software packages:
  - npu-firmware: firmware installation package
  - npu-driver: driver installation package
  - nnrt: offline inference engine package. It contains the ACL library (ACLlib), which is used for model inference of applications. nnrt supports offline inference only. This software package is installed in the Atlas Intelligent Edge Solution scenario.
  - nnae: deep learning engine package. It contains the FWK library (Fwklib) and OPP. nnae supports offline and online inference (with TensorFlow and MindSpore frameworks only). This software package is installed in the Ascend Data Center Solution scenario.
  - tfplugin: plugin package, an adaptation plugin used to connect to the upper-layer framework TensorFlow. If the deep learning framework TensorFlow is used for online inference, you need to install the software package.
  - AI framework: Only TensorFlow and MindSpore frameworks are supported. Before performing online inference, install an AI framework.

  For details about the software packages, see **11 Software Packages**.
### 3.5.2 Preparing the Hardware Environment

Prepare the hardware environment by referring to the *Atlas 300I Inference Card User Guide (Model 3010).*

Table 3-3 lists the OSs supported by an Atlas 300I inference card (model 3010).

<table>
<thead>
<tr>
<th>OS</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ubuntu</td>
<td>OS version: Ubuntu 18.04.1 Processor architecture: x86_64/AArch64</td>
<td>Official Ubuntu website</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x86_64</td>
</tr>
<tr>
<td></td>
<td></td>
<td>aarch64</td>
</tr>
<tr>
<td>OS</td>
<td>Version</td>
<td>How to Obtain</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CentOS</td>
<td>OS version: CentOS 7.6</td>
<td>Official CentOS website</td>
</tr>
<tr>
<td></td>
<td>Processor architecture: x86_64/AArch64</td>
<td>• x86_64 Download the required version from <a href="http://vault.centos.org/7.6.1810/isos/x86_64/">http://vault.centos.org/7.6.1810/isos/x86_64/</a>. The recommended version is CentOS-7-x86_64-DVD-1810.iso.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• aarch64 Download the recommended version CentOS-7-aarch64-Everything-1810.iso from <a href="https://archive.kernel.org/centos-vault/altarch/7.6.1810/isos/aarch64/">https://archive.kernel.org/centos-vault/altarch/7.6.1810/isos/aarch64/</a>.</td>
</tr>
<tr>
<td>CentOS</td>
<td>OS version: CentOS 8.2</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Processor architecture: x86_64/AArch64</td>
<td>-</td>
</tr>
<tr>
<td>EulerOS</td>
<td>OS version: EulerOS 2.8</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Processor architecture: AArch64</td>
<td>-</td>
</tr>
<tr>
<td>EulerOS</td>
<td>OS version: EulerOS 2.9</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Processor architecture: x86_64/AArch64</td>
<td>-</td>
</tr>
<tr>
<td>Kylin</td>
<td>OS version: Kylin V10 SP1</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Processor architecture: AArch64</td>
<td>-</td>
</tr>
<tr>
<td>Ubuntu</td>
<td>OS version: Ubuntu 16.04.3</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Processor architecture: x86_64</td>
<td>-</td>
</tr>
<tr>
<td>SUSE</td>
<td>OS version: SLES 12 SP4</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Processor architecture: x86_64</td>
<td>-</td>
</tr>
<tr>
<td>EulerOS</td>
<td>OS version: EulerOS 2.5</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Processor architecture: x86_64</td>
<td>-</td>
</tr>
<tr>
<td>BC-Linux</td>
<td>OS version: BC-Linux 7.6</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Processor architecture: x86_64</td>
<td>-</td>
</tr>
<tr>
<td>OS</td>
<td>Version</td>
<td>How to Obtain</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Kylin</td>
<td>OS version: Kylin V10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Processor architecture:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AArch64</td>
<td></td>
</tr>
<tr>
<td>Tlinux</td>
<td>OS version: Tlinux 2.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Processor architecture:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AArch64</td>
<td></td>
</tr>
<tr>
<td>NeoKylin</td>
<td>OS version: NeoKylin 7.6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Processor architecture:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AArch64</td>
<td></td>
</tr>
<tr>
<td>UOS</td>
<td>OS version: UOS V20 SP1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Processor architecture:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AArch64</td>
<td></td>
</tr>
<tr>
<td>SLES</td>
<td>OS version: SLES 12 SP5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Processor architecture:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>x86_64</td>
<td></td>
</tr>
</tbody>
</table>

### 3.6 A300T-9000

#### 3.6.1 Installation Scheme

The installation for the Atlas 300T training card (model 9000) development or operating environment is as follows:

- Development environment

  *Figure 3-9* shows the software to be installed in the development environment.

  Software packages:
  - npu-firmware: firmware installation package
  - npu-driver: driver installation package
  - Toolkit: development kit. It is used for application development, operator customization, and model conversion. A development kit contains the library files required for developing applications and auxiliary development tools such as the Ascend Tensor Compiler (ATC) for model conversion.
  - tfplugin: plugin package, an adaptation plugin used to connect to the upper-layer framework TensorFlow. If the deep learning framework TensorFlow is used for online inference or training, you need to install the software package.
  - AI framework: deep learning framework. For example, MindSpore, TensorFlow, or PyTorch.

For details about the software packages, see 11 Software Packages.
Operating environment

**Figure 3-10** shows the software to be installed in the operating environment. Software packages:

- npu-firmware: firmware installation package
- npu-driver: driver installation package
- nnae: deep learning engine package. It contains the FWK library (Fwklib) and operator library OPP component. Offline inference, online inference, and training are supported.
- tfplugin: plugin package, an adaptation plugin used to connect to the upper-layer framework TensorFlow. If the deep learning framework TensorFlow is used for online inference or training, you need to install the software package.
- AI framework: deep learning framework. For example, MindSpore, TensorFlow, or PyTorch.

For details about the software packages, see [11 Software Packages](#).
• Offline inference: converts models into OM models based on the original AI framework and does not depend on AI frameworks for inference.
• Online inference: ports applications that use the original AI framework for inference to Ascend AI Processors. In this scenario, AI frameworks are needed for inference.

### 3.6.2 Preparing the Hardware Environment

Prepare the hardware environment by referring to the *Atlas 300T AI Training Card User Guide (Model 9000)*.

**Table 3-4** lists the OSs supported by training servers configured with the Atlas 300T AI training cards (model 9000).

**Table 3-4 Supported OSs**

<table>
<thead>
<tr>
<th>OS</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ubuntu</td>
<td>OS version: Ubuntu 18.04</td>
<td>Official Ubuntu website</td>
</tr>
<tr>
<td></td>
<td>Processor architecture: x86_64/AArch64</td>
<td>• x86_64 Download the recommended version ubuntu-18.04.1-server-amd64.iso from <a href="http://old-releases.ubuntu.com/releases/18.04.1/">http://old-releases.ubuntu.com/releases/18.04.1/</a>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• aarch64 Download the recommended version ubuntu-18.04.1-server-arm64.iso from <a href="http://old-releases.ubuntu.com/releases/18.04.1/">http://old-releases.ubuntu.com/releases/18.04.1/</a>.</td>
</tr>
<tr>
<td>CentOS</td>
<td>OS version: CentOS 7.6</td>
<td>Official CentOS website</td>
</tr>
<tr>
<td></td>
<td>Processor architecture: x86_64/AArch64</td>
<td>• x86_64 Download the required version from <a href="http://vault.centos.org/7.6.1810/ios/x86_64/">http://vault.centos.org/7.6.1810/ios/x86_64/</a>. The recommended version is CentOS-7-x86_64-DVD-1810.iso.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• aarch64 Download the recommended version CentOS-7-aarch64-Everything-1810.iso from <a href="https://archive.kernel.org/centos-vault/altarch/7.6.1810/isos/aarch64/">https://archive.kernel.org/centos-vault/altarch/7.6.1810/isos/aarch64/</a>.</td>
</tr>
<tr>
<td>CentOS</td>
<td>OS version: CentOS 8.2</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Processor architecture: x86_64/AArch64</td>
<td>-</td>
</tr>
</tbody>
</table>
### 3.7 A500-3000 and A500-3010

#### 3.7.1 Installation Scheme

If the content in this document is not specific to any model, the product is described as an Atlas 500 AI edge station. The Atlas 500 AI Edge Station can only be used as the operating environment.

- **Development environment**
  
  You need to prepare another server to set up the development environment. For details, see [3.1 Non-Ascend Devices](#).

- **Operating environment**
  
  - An Atlas 500 AI edge station is preinstalled with the EulerOS, driver, and firmware. For details about the initial configuration of the Atlas 500 AI edge station, see "Installation and Configuration > Initial Configuration" in the [Atlas 500 AI Edge Station User Guide (Models 3000, 3010)](#).

  **Figure 3-11** shows the installation scheme.

  For details about the software packages, see [11 Software Packages](#).

**Figure 3-11** Atlas 500 AI edge station

<table>
<thead>
<tr>
<th>OS</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>EulerOS</td>
<td>OS version: EulerOS 2.8</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Processor architecture:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AArch64</td>
<td></td>
</tr>
<tr>
<td>Kylin</td>
<td>OS version: Kylin V10 SP1</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Processor architecture:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>x86_64/AArch64</td>
<td></td>
</tr>
<tr>
<td>Debian</td>
<td>OS version: Debian 9.9</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Processor architecture:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>x86_64</td>
<td></td>
</tr>
</tbody>
</table>
3.8 A500 Pro-3000

3.8.1 Installation Scheme

The installation for the Atlas 500 Pro intelligent edge server (model 3000) development or operating environment is as follows:

- **Development environment**
  
  Figure 3-12 shows the software to be installed in the development environment.

  Software packages:
  
  - npu-firmware: firmware installation package
  - npu-driver: driver installation package
  - Toolkit: development kit. It is used for application development, operator customization, and model conversion. A development kit contains the library files required for developing applications and auxiliary development tools such as the Ascend Tensor Compiler (ATC) for model conversion.

  For details about the software packages, see 11 Software Packages.

  ![Development environment](image)

- **Operating environment**

  Figure 3-13 shows the software to be installed in the operating environment.

  Software packages:
  
  - npu-firmware: firmware installation package
  - npu-driver: driver installation package
  - nnrt: offline inference engine package. It contains the ACL library (ACLlib), which is used for model inference for applications. Only offline inference is supported.

  For details about the software packages, see 11 Software Packages.
3.8.2 Preparing the Hardware Environment

For details about how to install an Atlas 500 Pro AI edge server (model 3000), set basic server parameters, and install the OS, see the *Atlas 500 Pro AI Edge Server User Guide (Model 3000)*. After the OS is installed, configure an IP address for the service network port. For details, see 10.2 Configuring the NIC IP Address.

Table 3-5 lists the OSs supported by the Atlas 500 Pro AI edge server (model 3000).

**Table 3-5 Supported OSs**

<table>
<thead>
<tr>
<th>OS</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ubuntu</td>
<td>OS version: Ubuntu 18.04.1</td>
<td>Official Ubuntu website</td>
</tr>
<tr>
<td></td>
<td>Processor architecture:</td>
<td>Download the recommended version</td>
</tr>
<tr>
<td></td>
<td>AArch64</td>
<td><a href="http://old-releases.ubuntu.com/releases/18.04.1/">ubuntu-18.04.1-server-arm64.iso</a></td>
</tr>
<tr>
<td>EulerOS</td>
<td>OS version: EulerOS 2.8</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Processor architecture:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AArch64</td>
<td></td>
</tr>
<tr>
<td>CentOS</td>
<td>OS version: CentOS 7.6</td>
<td>Official CentOS website</td>
</tr>
<tr>
<td></td>
<td>Processor architecture:</td>
<td>Download the recommended version</td>
</tr>
<tr>
<td></td>
<td>AArch64</td>
<td><a href="https://archive.kernel.org/centos-vault/altarch/7.6.1810/isos/aarch64/">CentOS-7-aarch64-Everything-1810.iso</a>.</td>
</tr>
</tbody>
</table>

**NOTE**

- Offline inference: converts models into OM models based on the original AI framework and does not depend on AI frameworks for inference.
- Online inference: ports applications that use the original AI framework for inference to Ascend AI Processors. In this scenario, AI frameworks are needed for inference.
### 3.9 A800-3000

#### 3.9.1 Installation Scheme

The installation for the Atlas 800 inference server (model 3000) development or operating environment is as follows:

- Development environment
  - **Figure 3-14** shows the software to be installed in the development environment.
  - **Software packages:**
    - npu-firmware: firmware installation package
    - npu-driver: driver installation package
    - Toolkit: development kit. It is used for application development, operator customization, and model conversion. A development kit contains the library files required for developing applications and auxiliary development tools such as the Ascend Tensor Compiler (ATC) for model conversion.
    - tfplugin: plugin package, an adaptation plugin used to connect to the upper-layer framework TensorFlow. If the deep learning framework TensorFlow is used for online inference, you need to install the software package.
    - AI framework: Only TensorFlow and MindSpore frameworks are supported. Before performing online inference, install an AI framework.

For details about the software packages, see 11 Software Packages.
Operating environment

Figure 3-15 shows the software to be installed in the operating environment. nnae supports offline and online inference.

Software packages:
- npu-firmware: firmware installation package
- npu-driver: driver installation package
- nnae: deep learning engine package. It contains the FWK library (Fwklib) and OPP. nnae supports offline inference and online inference (with TensorFlow and MindSpore frameworks only).
- tfplugin: plugin package, an adaptation plugin used to connect to the upper-layer framework TensorFlow. If the deep learning framework TensorFlow is used for online inference, you need to install the software package.
- AI framework: Only TensorFlow and MindSpore frameworks are supported. Before performing online inference, install an AI framework.

For details about the software packages, see 11 Software Packages.

Figure 3-15 Operating environment
### 3.9.2 Preparing the Hardware Environment

For details about how to install an Atlas 800 inference server (model 3000), set basic server parameters, and install the OS, see the *Atlas 800 Inference Server User Guide (Model 3000)*. After the OS is installed, configure an IP address for the service network port. For details, see 10.2 Configuring the NIC IP Address.

Table 3-6 lists the OSs supported by the Atlas 800 inference server (model 3000).

<table>
<thead>
<tr>
<th>OS</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
</table>
| Ubuntu | OS version: Ubuntu 18.04.1  
Processor architecture: AArch64 | Official Ubuntu website  
Download the recommended version ubuntu-18.04.1-server-arm64.iso from http://old-releases.ubuntu.com/releases/18.04.1/.
| CentOS | OS version: CentOS 7.6  
Processor architecture: AArch64 | Official CentOS website  
Download the recommended version CentOS-7-aarch64-Everything-1810.iso from https://archive.kernel.org/centos-vault/altarch/7.6.1810/isos/aarch64/.
| CentOS | OS version: CentOS 8.2  
Processor architecture: AArch64 | - |
| EulerOS | OS version: EulerOS 2.8  
Processor architecture: AArch64 | - |
| EulerOS | OS version: EulerOS 2.9  
Processor architecture: AArch64 | - |
| Kylin | OS version: Kylin V10 SP1  
Processor architecture: AArch64 | - |
3.10 A800-3010

3.10.1 Installation Scheme

The installation for the Atlas 800 inference server (model 3010) development or operating environment is as follows:

- Development environment

  Figure 3-16 shows the software to be installed in the development environment.

  Software packages:
  - npu-firmware: firmware installation package
  - npu-driver: driver installation package
  - Toolkit: development kit. It is used for application development, operator customization, and model conversion. A development kit contains the library files required for developing applications and auxiliary development tools such as the Ascend Tensor Compiler (ATC) for model conversion.
  - tfplugin: plugin package, an adaptation plugin used to connect to the upper-layer framework TensorFlow. If the deep learning framework TensorFlow is used for online inference, you need to install the software package.
  - AI framework: Only TensorFlow and MindSpore frameworks are supported. Before performing online inference, install an AI framework.

  For details about the software packages, see 11 Software Packages.

  Figure 3-16 Development environment

- Operating environment

  Figure 3-17 shows the software to be installed in the operating environment.

  Software packages:
- npu-firmware: firmware installation package
- npu-driver: driver installation package
- nnae: deep learning engine package. It contains the FWK library (Fwklib) and OPP. nnae supports offline inference and online inference (with TensorFlow and MindSpore frameworks only).
- tfplugin: plugin package, an adaptation plugin used to connect to the upper-layer framework TensorFlow. If the deep learning framework TensorFlow is used for online inference, you need to install the software package.
- AI framework: Only TensorFlow and MindSpore frameworks are supported. Before performing online inference, install an AI framework.

For details about the software packages, see 11 Software Packages.

Figure 3-17 Operating environment

NOTE
- Offline inference: converts models into OM models based on the original AI framework and does not depend on AI frameworks for inference.
- Online inference: ports applications that use the original AI framework for inference to Ascend AI Processors. In this scenario, AI frameworks are needed for inference.

3.10.2 Preparing the Hardware Environment

For details about how to install an Atlas 800 inference server (model 3010), set basic server parameters, and install the OS, see the Atlas 800 Inference Server User Guide (Model 3010). After the OS is installed, configure an IP address for the service network port. For details, see 10.2 Configuring the NIC IP Address.

Table 3-7 lists the OSs supported by the Atlas 800 inference server (model 3010).
Table 3-7 Supported OSs

<table>
<thead>
<tr>
<th>OS</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ubuntu</td>
<td>OS version: Ubuntu 18.04.1 Processor architecture: x86_64</td>
<td>Official Ubuntu website Download the recommended version ubuntu-18.04.1-server-amd64.iso from <a href="http://old-releases.ubuntu.com/releases/18.04.1/">http://old-releases.ubuntu.com/releases/18.04.1/</a></td>
</tr>
<tr>
<td>CentOS</td>
<td>OS version: CentOS 7.6 Processor architecture: x86_64</td>
<td>Official CentOS website Download the recommended version CentOS-7-x86_64-DVD-1810.iso from <a href="http://vault.centos.org/7.6.1810/isos/x86_64/">http://vault.centos.org/7.6.1810/isos/x86_64/</a></td>
</tr>
<tr>
<td>CentOS</td>
<td>OS version: CentOS 8.2 Processor architecture: x86_64</td>
<td>-</td>
</tr>
<tr>
<td>EulerOS</td>
<td>OS version: EulerOS 2.9 Processor architecture: x86_64</td>
<td>-</td>
</tr>
<tr>
<td>BC-Linux</td>
<td>OS version: BC-Linux 7.6 Processor architecture: x86_64</td>
<td>-</td>
</tr>
<tr>
<td>SLES</td>
<td>OS version: SLES 12 SP5 Processor architecture: x86_64</td>
<td>-</td>
</tr>
</tbody>
</table>

3.11 A800-9000

3.11.1 Installation Scheme

The installation for the Atlas 800 training server (model 9000) development or operating environment is as follows:

- Development environment

  Figure 3-18 shows the software to be installed in the development environment.

  Software packages:
  - npu-firmware: firmware installation package
  - npu-driver: driver installation package
  - Toolkit: development kit. It is used for application development, operator customization, and model conversion. A development kit contains the
library files required for developing applications and auxiliary
development tools such as the Ascend Tensor Compiler (ATC) for model
conversion.
- tfplugin: plugin package, an adaptation plugin used to connect to the
upper-layer framework TensorFlow. If the deep learning framework
TensorFlow is used for online inference or training, you need to install the
software package.
- AI framework: deep learning framework. For example, MindSpore,
TensorFlow, or PyTorch.
For details about the software packages, see 11 Software Packages.

**Figure 3-18 Development environment**

- Operating environment
**Figure 3-19** shows the software to be installed in the operating environment.
Software packages:
  - npu-firmware: firmware installation package
  - npu-driver: driver installation package
  - nnae: deep learning engine package. It contains the FWK library (Fwklib)
and operator library OPP component. Offline inference, online inference,
and training are supported.
  - tfplugin: plugin package, an adaptation plugin used to connect to the
upper-layer framework TensorFlow. If the deep learning framework
TensorFlow is used for online inference or training, you need to install the
software package.
  - AI framework: deep learning framework. For example, MindSpore,
TensorFlow, or PyTorch.
For details about the software packages, see 11 Software Packages.
**Figure 3-19** Operating environment

**NOTE**

- Offline inference: converts models into OM models based on the original AI framework and does not depend on AI frameworks for inference.
- Online inference: ports applications that use the original AI framework for inference to Ascend AI Processors. In this scenario, AI frameworks are needed for inference.

### 3.11.2 Preparing the Hardware Environment

For details about how to install an Atlas 800 training server (model 9000), set basic server parameters, and install the OS, see the Atlas 800 Training Server User Guide (Model 9000, Air Cooling) or Atlas 800 Training Server User Guide (Model 9000, Liquid Cooling).

Table 3-8 lists the OSs supported by the Atlas 800 training server (model 9000). If CentOS is used, set One Numa Per Socket to Enabled on the BIOS screen. For details, see "Preparations > Configuring the BIOS" in the Atlas 800 AI Training Server CentOS 7.6 Installation Guide (Model 9000).

**Table 3-8** Supported OSs

<table>
<thead>
<tr>
<th>OS</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>CentOS</td>
<td>OS version: CentOS 7.6 Processor architecture: AArch64</td>
<td>Official CentOS website Download the following recommended version from <a href="https://archive.kernel.org/centos-vault/altarch/7.6.1810/isos/aarch64/">https://archive.kernel.org/centos-vault/altarch/7.6.1810/isos/aarch64/</a>: CentOS-7-aarch64-Everything-1810.iso.</td>
</tr>
<tr>
<td>OS</td>
<td>Version</td>
<td>How to Obtain</td>
</tr>
<tr>
<td>----------</td>
<td>----------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>CentOS</td>
<td>OS version: CentOS 8.2</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Processor architecture: aarch64</td>
<td></td>
</tr>
<tr>
<td>EulerOS</td>
<td>OS version: EulerOS 2.8</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Processor architecture: AArch64</td>
<td></td>
</tr>
<tr>
<td>Kylin</td>
<td>OS version: Kylin V10 SP1</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Processor architecture: AArch64</td>
<td></td>
</tr>
<tr>
<td>BC-Linux</td>
<td>OS version: BC-Linux 7.6</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Processor architecture: AArch64</td>
<td></td>
</tr>
<tr>
<td>BC-Linux</td>
<td>OS version: BC-Linux 7.7</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Processor architecture: AArch64</td>
<td></td>
</tr>
</tbody>
</table>

### 3.12 A800-9010

#### 3.12.1 Installation Scheme

The installation for the Atlas 800 training server (model 9010) development or operating environment is as follows:

- Development environment
  - **Figure 3-20** shows the software to be installed in the development environment.
  - Software packages:
    - npu-firmware: firmware installation package
    - npu-driver: driver installation package
    - Toolkit: development kit. It is used for application development, operator customization, and model conversion. A development kit contains the library files required for developing applications and auxiliary development tools such as the Ascend Tensor Compiler (ATC) for model conversion.
    - tfplugin: plugin package, an adaptation plugin used to connect to the upper-layer framework TensorFlow. If the deep learning framework TensorFlow is used for online inference or training, you need to install the software package.
    - AI framework: deep learning framework. For example, MindSpore, TensorFlow, or PyTorch.
For details about the software packages, see **11 Software Packages**.

**Figure 3-20** Development environment

- Operating environment

*Figure 3-21* shows the software to be installed in the operating environment.

Software packages:

- npu-firmware: firmware installation package
- npu-driver: driver installation package
- nnae: deep learning engine package. It contains the FWK library (Fwklib) and operator library OPP component. Offline inference, online inference, and training are supported.
- tfplugin: plugin package, an adaptation plugin used to connect to the upper-layer framework TensorFlow. If the deep learning framework TensorFlow is used for online inference or training, you need to install the software package.
- AI framework: deep learning framework. For example, MindSpore, TensorFlow, or PyTorch.

For details about the software packages, see **11 Software Packages**.
### 3.12.2 Preparing the Hardware Environment

For details about how to install an Atlas 800 training server (model 9010), set basic server parameters, and install the OS, see the *Atlas 800 AI Training Server User Guide (Model 9010)*.

Table 3-9 lists the OSs supported by the Atlas 800 training server (model 9010).

#### Table 3-9 Supported OSs

<table>
<thead>
<tr>
<th>OS</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ubuntu</td>
<td>OS version: Ubuntu 18.04.1</td>
<td>Official Ubuntu website</td>
</tr>
<tr>
<td></td>
<td>Processor architecture: x86_64</td>
<td>Download the recommended version <em>ubuntu-18.04.1-server-amd64.iso</em> from</td>
</tr>
<tr>
<td></td>
<td></td>
<td><em><a href="http://old-releases.ubuntu.com/releases/18.04.1/">http://old-releases.ubuntu.com/releases/18.04.1/</a>.</em></td>
</tr>
<tr>
<td>CentOS</td>
<td>OS version: CentOS 7.6</td>
<td>Official CentOS website</td>
</tr>
<tr>
<td></td>
<td>Processor architecture: x86_64</td>
<td>Download the required version from <em><a href="http://vault.centos.org/7.6.1810/isos/x86_64/">http://vault.centos.org/7.6.1810/isos/x86_64/</a></em>. The recommended version is CentOS-7-x86_64-DVD-1810.iso.</td>
</tr>
<tr>
<td>CentOS</td>
<td>OS version: CentOS 8.2</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Processor architecture: x86_64</td>
<td></td>
</tr>
<tr>
<td>Kylin</td>
<td>OS version: Kylin V10 SP1</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Processor architecture: x86_64</td>
<td></td>
</tr>
<tr>
<td>Debian</td>
<td>OS version: Debian 9.9</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Processor architecture: x86_64</td>
<td></td>
</tr>
<tr>
<td>Debian</td>
<td>OS version: Debian 10.0</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Processor architecture: x86_64</td>
<td></td>
</tr>
</tbody>
</table>
### 3.13 A900-9000

#### 3.13.1 Installation Scheme

The installation for the Atlas 900 AI cluster (model 9000) development or operating environment is as follows:

- Development environment
  
  *Figure 3-22* shows the software to be installed in the development environment.

  **Software packages:**
  - `npu-firmware`: firmware installation package
  - `npu-driver`: driver installation package
  - `Toolkit`: development kit. It is used for application development, operator customization, and model conversion. A development kit contains the library files required for developing applications and auxiliary development tools such as the Ascend Tensor Compiler (ATC) for model conversion.
  - `tfplugin`: plugin package, an adaptation plugin used to connect to the upper-layer framework TensorFlow. If the deep learning framework TensorFlow is used for online inference or training, you need to install the software package.
  - `AI framework`: deep learning framework. For example, MindSpore, TensorFlow, or PyTorch.

  For details about the software packages, see [11 Software Packages](#).
Operating environment

**Figure 3-23** shows the software to be installed in the operating environment. Software packages:

- **npu-firmware**: firmware installation package
- **npu-driver**: driver installation package
- **nnae**: deep learning engine package. It contains the FWK library (Fwklib) and operator library OPP component. Offline inference, online inference, and training are supported.
- **tfplugin**: plugin package, an adaptation plugin used to connect to the upper-layer framework TensorFlow. If the deep learning framework TensorFlow is used for online inference or training, you need to install the software package.
- **AI framework**: deep learning framework. For example, MindSpore, TensorFlow, or PyTorch.

For details about the software packages, see [11 Software Packages](#).
3.13.2 Preparing the Hardware Environment

For details about how to install an Atlas 900 AI cluster, set basic server parameters, and install the OS, see the corresponding document based on the cluster configuration.

- Atlas 900 PoD User Guide (Model 9000, DC)
- Atlas 900 PoD User Guide (Model 9000, AC)
- Atlas 900 Compute Node User Guide (Liquid Cooling)

Table 3-10 lists the OSs supported by the Atlas 900 AI cluster.

<table>
<thead>
<tr>
<th>OS</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
</table>
| CentOS     | OS version: CentOS 7.6 Processor architecture: AArch64                    | Official CentOS website Download the following recommended version from https://archive.kernel.org/centos-vault/altarch/7.6.1810/isos/aarch64/: CentOS-7-aarch64-Everything-1810.iso.
| CentOS     | OS version: CentOS 8.2 Processor architecture: aarch64                   | -                                                                             |
| EulerOS    | OS version: EulerOS 2.8 Processor architecture: AArch64                  | -                                                                             |
| Kylin      | OS version: Kylin V10 SP1 Processor architecture: AArch64                | -                                                                             |
| BC-Linux   | OS version: BC-Linux 7.6 Processor architecture: AArch64                 | -                                                                             |
4 Installing the Driver and Firmware

Perform the installation by referring to Table 4-1.

**NOTE**

For initial installation, install the driver and then the firmware; for overwrite installation, install the firmware and then the driver.

<table>
<thead>
<tr>
<th>Product Model</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atlas 200 AI accelerator module (model 3000)</td>
<td>Atlas 200 AI Accelerator Module 1.0.7 or Later Software Installation and Maintenance Guide (EP, Model 3000)</td>
</tr>
<tr>
<td>Atlas 300I inference card (model 3000)</td>
<td>&quot;Installation and Maintenance&quot; in the Atlas 300I Inference Card User Guide (Model 3000)</td>
</tr>
<tr>
<td>Atlas 300I inference card (model 3010)</td>
<td>&quot;Installation and Maintenance&quot; in the Atlas 300I Inference Card User Guide (Model 3010)</td>
</tr>
<tr>
<td>Atlas 300T training card (model 9000)</td>
<td>&quot;Installation and Maintenance&quot; in the Atlas 300T Training Card Driver and Firmware Installation and Upgrade Guide (Model 9000)</td>
</tr>
<tr>
<td>Atlas 500 Pro AI edge server (model 3000)</td>
<td>&quot;Installation and Maintenance&quot; in the Atlas 300I Inference Card User Guide (Model 3000)</td>
</tr>
<tr>
<td>Atlas 800 inference server (model 3000)</td>
<td>&quot;Installation and Maintenance&quot; in the Atlas 300I Inference Card User Guide (Model 3000)</td>
</tr>
<tr>
<td>Atlas 800 inference server (model 3010)</td>
<td>&quot;Installation and Maintenance&quot; in the Atlas 300I Inference Card User Guide (Model 3010)</td>
</tr>
<tr>
<td>Atlas 800 training server (model 9000)</td>
<td>Atlas 800 AI Training Server Driver and Firmware Installation and Upgrade Guide (Model 9000)</td>
</tr>
<tr>
<td>Product Model</td>
<td>Reference</td>
</tr>
<tr>
<td>---------------------------------------------------</td>
<td>----------------------------------------------------------------</td>
</tr>
<tr>
<td>Atlas 800 training server (model 9010)</td>
<td><em>Atlas 800 AI Training Server Driver and Firmware Installation and Upgrade Guide (Model 9010)</em></td>
</tr>
<tr>
<td>Atlas 900 AI cluster (model 9000)</td>
<td><em>Atlas 900 Compute Node Driver and Firmware Installation and Upgrade Guide</em></td>
</tr>
</tbody>
</table>
5 Installing the Development Environment

5.1 Obtaining Software Packages

Downloading Software Packages

Before installing the software, obtain required software packages by referring to Table 5-1. The versions of the software packages must be consistent.
Table 5-1 CANN software packages

<table>
<thead>
<tr>
<th>Name</th>
<th>Package</th>
<th>Description</th>
<th>How to Obtain</th>
</tr>
</thead>
</table>
| Development kit       | Ascend-cann-toolkit_{version}_linux-{arch}.run                           | ● It is used for application development, operator customization, and model conversion. The development kit contains the library files required for developing applications and auxiliary development tools such as the ATC for model conversion.  
● Obtain the software packages based on the CPU architecture (x86_64 or AArch64).  
● If the operating environment is AArch64 and the development environment is x86_64, you need to obtain the development kits of both architectures. | Link          |
| Framework plugin      | Ascend-cann-tfplugin_{version}_linux-{arch}.run                          | (Optional) Plugin package, an adaptation plugin used to connect to the upper-layer framework TensorFlow.  
If the deep learning framework TensorFlow is used for online inference or training, you need to obtain the software package. | Link          |
Checking the Integrity of Software Packages

To prevent a software package from being maliciously tampered with during transmission or storage, download the corresponding verification file for integrity verification when downloading the software package.

If the extension of the downloaded verification file is .sha256sum, perform the following steps:

Place the downloaded software package and its verification file in the same directory and run the following command as the root user:

```
sha256sum -c Ascend-cann-nnrt_{software version}_linux-aarch64.run.sha256sum
```

The file passes the integrity verification if you see information similar to the following:

```
Ascend-cann-nnrt_{software version}_linux-aarch64.run: OK
```

5.2 Preparing Installation and Running Users

Checking the umask of the root User

1. Log in to the installation environment as the root user.
2. Check the umask value of the root user.

```
umask
```

3. If the umask value is not 0022, append `umask 0022` to the file and save the file.
   a. Run the following command in any directory to open the .bashrc file.

```
vi ~/.bashrc
```

Append `umask 0022` to the end of the file.
   b. Run the `:wq!` command to save the file and exit.
   c. Run the `source ~/.bashrc` command for the modification to take effect immediately.

Creating Installation and Running Users

- Running user: user who runs the inference service or performs training.
- Installation user: user who installs the software package.
  - If the root user is used for installation, you are advised to use a non-root user to run related services. Before the installation, create a running user.
  - If a non-root user is used for installation, the installation and running users must be the same.
- For installation as the root user:
(Recommended) If the created running user is **HwHiAiUser**, which is the default running user of the CANN software package, you do not need to specify the running user during installation.

- If the created running user is not **HwHiAiUser**, you need to specify the running user when installing the CANN software package.

### For installation as a non-root user:

- If a non-root user exists, you do not need to create one.
- To create a non-root user, perform the following steps:

  **NOTE**

  - If the CANN software package is installed by a non-root user, ensure that the owner group of the non-root user is the same as that of the user who runs the driver. If they are different, add the non-root user to the group of the driver running user.
  - The created running user cannot belong to the root user group because security risks may exist.

Run the following commands as the **root** user to create a non-root user.

1. Create a non-root user.

   ```bash
   groupadd usergroup
   useradd -g usergroup -d /home/username -m username -s /bin/bash
   ```

   Create the running user **HwHiAiUser**.

   ```bash
   groupadd HwHiAiUser
   useradd -g HwHiAiUser -d /home/HwHiAiUser -m HwHiAiUser -s /bin/bash
   ```

2. Set the password for the non-root user.

   ```bash
   passwd username
   ```

   Example:

   ```bash
   passwd HwHiAiUser
   ```

  **NOTE**

  - After the **HwHiAiUser** user is created, do not disable the login authentication function of the user.
  - The password validity period is 90 days. You can change the validity period in the **/etc/login.defs** file or using the **chage** command. For details, see **10.8 Setting User Account Validity Period**.

### 5.3 Installing OS Dependencies

#### 5.3.1 Before You Start

Before installing the CANN software, install related dependencies.

This section describes how to install the dependencies on Ubuntu 18.04 and CentOS 7.6. You can refer to the following procedure to install the dependencies on other OSs.

#### 5.3.2 Dependency List

- Ubuntu 18.04
<table>
<thead>
<tr>
<th>Type</th>
<th>Version Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>3.7.0–3.7.9</td>
</tr>
<tr>
<td>cmake</td>
<td>≥ 3.5.1</td>
</tr>
<tr>
<td>make</td>
<td>-</td>
</tr>
<tr>
<td>gcc</td>
<td>• Offline inference</td>
</tr>
<tr>
<td></td>
<td>The GCC version must be 4.8.5 or later.</td>
</tr>
<tr>
<td></td>
<td>• Online inference, training, and Ascend graph development</td>
</tr>
<tr>
<td></td>
<td>The GCC version must be 7.3.0 or later.</td>
</tr>
<tr>
<td></td>
<td>If the GCC version is earlier than 7.3.0, install the GCC by referring to 10.6</td>
</tr>
<tr>
<td></td>
<td>Installing GCC 7.3.0.</td>
</tr>
<tr>
<td>g++</td>
<td>There is no version requirement. The version to be installed is subject to the source provided by the OS.</td>
</tr>
<tr>
<td>zlib1g</td>
<td>≥1.13.3 &amp;&amp; &lt;1.20</td>
</tr>
<tr>
<td>zlib1g-dev</td>
<td></td>
</tr>
<tr>
<td>libsqlite3-dev</td>
<td></td>
</tr>
<tr>
<td>openssl</td>
<td></td>
</tr>
<tr>
<td>libssl-dev</td>
<td></td>
</tr>
<tr>
<td>libffi-dev</td>
<td></td>
</tr>
<tr>
<td>unzip</td>
<td></td>
</tr>
<tr>
<td>pciutils</td>
<td></td>
</tr>
<tr>
<td>net-tools</td>
<td></td>
</tr>
<tr>
<td>libblas-dev</td>
<td></td>
</tr>
<tr>
<td>gfortran</td>
<td></td>
</tr>
<tr>
<td>libblas3</td>
<td></td>
</tr>
<tr>
<td>liblapack-dev</td>
<td></td>
</tr>
<tr>
<td>liblapack3</td>
<td></td>
</tr>
<tr>
<td>libopenblas-dev</td>
<td></td>
</tr>
<tr>
<td>numpy</td>
<td>&gt;=1.13.3 &amp;&amp; &lt;1.20</td>
</tr>
<tr>
<td>decorator</td>
<td>&gt;=4.4.0</td>
</tr>
<tr>
<td>sympy</td>
<td>&gt;=1.4</td>
</tr>
<tr>
<td>cffi</td>
<td>&gt;=1.12.3</td>
</tr>
<tr>
<td>protobuf</td>
<td>&gt;=3.11.3</td>
</tr>
<tr>
<td>Type</td>
<td>Version Requirement</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>attrs</td>
<td>There is no version requirement. The version is subject to the pip source.</td>
</tr>
<tr>
<td>pyyaml</td>
<td></td>
</tr>
<tr>
<td>pathlib2</td>
<td></td>
</tr>
<tr>
<td>scipy</td>
<td></td>
</tr>
<tr>
<td>requests</td>
<td></td>
</tr>
<tr>
<td>psutil</td>
<td></td>
</tr>
</tbody>
</table>

- CentOS 7.6

Table 5-3 Dependency information

<table>
<thead>
<tr>
<th>Type</th>
<th>Version Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>3.7.0–3.7.9</td>
</tr>
<tr>
<td>cmake</td>
<td>≥ 3.5.1</td>
</tr>
<tr>
<td>make</td>
<td>-</td>
</tr>
<tr>
<td>gcc</td>
<td></td>
</tr>
<tr>
<td>gcc-c++</td>
<td></td>
</tr>
<tr>
<td>unzip</td>
<td></td>
</tr>
<tr>
<td>zlib-devel</td>
<td></td>
</tr>
<tr>
<td>libffi-devel</td>
<td></td>
</tr>
<tr>
<td>openssl-devel</td>
<td></td>
</tr>
<tr>
<td>pciutils</td>
<td></td>
</tr>
<tr>
<td>net-tools</td>
<td></td>
</tr>
<tr>
<td>sqlite-devel</td>
<td></td>
</tr>
<tr>
<td>lapack-devel</td>
<td></td>
</tr>
<tr>
<td>openblas-devel</td>
<td></td>
</tr>
<tr>
<td>gcc-gfortran</td>
<td></td>
</tr>
<tr>
<td>numpy</td>
<td>&gt;=1.13.3 &amp;&amp; &lt;1.20</td>
</tr>
<tr>
<td>decorator</td>
<td>&gt;=4.4.0</td>
</tr>
<tr>
<td>sympy</td>
<td>&gt;=1.4</td>
</tr>
<tr>
<td>cffi</td>
<td>&gt;=1.12.3</td>
</tr>
<tr>
<td>protobuf</td>
<td>&gt;=3.11.3</td>
</tr>
</tbody>
</table>

- Offline inference
  - The GCC version must be 4.8.5 or later.
- Online inference, training, and Ascend graph development
  - The GCC version must be 7.3.0 or later. If the GCC version is earlier than 7.3.0, install the GCC by referring to 10.6 Installing GCC 7.3.0.
<table>
<thead>
<tr>
<th>Type</th>
<th>Version Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>attrs</td>
<td>There is no version requirement. The version is subject to the pip source.</td>
</tr>
<tr>
<td>pyyaml</td>
<td></td>
</tr>
<tr>
<td>pathlib2</td>
<td></td>
</tr>
<tr>
<td>scipy</td>
<td></td>
</tr>
<tr>
<td>requests</td>
<td></td>
</tr>
<tr>
<td>psutil</td>
<td></td>
</tr>
</tbody>
</table>

### 5.3.3 Installation Procedure (Ubuntu 18.04)

#### Checking the Source Validity

The installation requires the download of related dependencies. Ensure that the installation environment can be connected to the network.

Run the following command as the `root` user to check whether the source is valid:

```bash
apt-get update
```

If an error is reported during command execution or dependency installation, check whether the network is connected, or replace the source in the `/etc/apt/sources.list` file with an available source or use an image source. For details about how to configure a network proxy, see 10.3 Configuring a System Network Proxy.

#### Configuring Permissions for the Installation User

Skip the following part if you install as the `root` user.

Perform the following steps as the `root` user:

1. Run the following command to install sudo:

   ```bash
   apt-get install sudo
   ```

2. Open the `/etc/sudoers` file:

   ```bash
   chmod u+w /etc/sudoers
   vi /etc/sudoers
   ```

3. Append the following content to the configuration file:

   ```
   username ALL=(ALL:ALL) NOPASSWD:SETENV:/usr/bin/apt-get, /usr/bin/pip, /bin/tar, /bin/mkdir, /bin/sh, /bin/bash, /usr/bin/make install, /bin/ln -s /usr/local/python3.7.5/bin/python /usr/local/python3.7.5/bin/python3.7.5, /bin/unzip
   ```

   In the preceding command, `username` indicates the installation user. Change it as required.
### CAUTION

- Ensure that the `/etc/sudoers` file contains `#includedir /etc/sudoers.d`. If the file does not contain `#includedir /etc/sudoers.d`, manually add it to the end of the file.
- After the CANN software package has been installed, you can cancel the sudo permission.
- When uninstalling or upgrading the CANN software package, you also need to configure the preceding user permissions.

4. Run `:wq!` to save the file.
5. Run the following command to revoke the write permission on the `/etc/sudoers` file:
   ```bash
   chmod u-w /etc/sudoers
   ```

### Installing Dependencies

#### Step 1  Check whether the Python dependencies and GCC software are installed.

Run the following commands to check whether the dependencies such as GCC, Make, and Python are installed:

```bash
gcc --version
g++ --version
make --version
cmake --version
dpkg -I zlib1g| grep zlib1g| grep ii
dpkg -I zlib1g-dev| grep zlib1g-dev| grep ii
dpkg -I libsqlite3-dev| grep libsqlite3-dev| grep ii
dpkg -I openssl| grep openssl| grep ii
dpkg -I libssl-dev| grep libssl-dev| grep ii
dpkg -I libffi-dev| grep libffi-dev| grep ii
dpkg -I unzip| grep unzip| grep ii
dpkg -I pciutils| grep pciutils| grep ii
dpkg -I net-tools| grep net-tools| grep ii
dpkg -I libblas-dev| grep libblas-dev| grep ii
dpkg -I gfortran| grep gfortran| grep ii
dpkg -I libblas3| grep libblas3| grep ii
dpkg -I libopenblas-dev| grep libopenblas-dev| grep ii
```

If the following information is displayed, the dependencies have been installed. Go to the next step. (The following information shows an example.)

```bash
gcc (Ubuntu 7.3.0-3ubuntu1~18.04) 7.3.0
GNU Make 4.7
cmake version 3.10.2
zlib1g:arm64 1:1.2.11.dfsg-0ubuntu2 arm64   compression library - runtime
zlib1g-dev:arm64 1:1.2.11.dfsg-0ubuntu2 arm64 compression library - development
libsqlite3-dev:arm64 3.22.0-1ubuntu0.3 arm64 SQLite 3 development files
openssl 1.1.1-1ubuntu2.1-18.04.6 arm64 Secure Sockets Layer toolkit - cryptographic utility
libssl-dev:arm64 1.1.1-1ubuntu2.1-18.04.6 arm64 Secure Sockets Layer toolkit - development files
libffi-dev:arm64 3.2.1-8 arm64 Foreign Function Interface library (development files)
unzip 6.0-21ubuntu1 arm64 De-archiver for .zip files
pciutils 1.3.5.2-1ubuntu1 arm64 Linux PCI Utilities
net-tools 1.60+git20161116.90da8a0-1ubuntu1 arm64 NET-3 networking toolkit
libblas-dev:arm64 3.7.1-4ubuntu1 arm64 Basic Linear Algebra Subroutines 3, static library
gfortran 4.7.4-0-1ubuntu2.3 arm64 GNU Fortran 95 compiler
libblas3:arm64 3.7.1-4ubuntu1 arm64 Basic Linear Algebra Reference implementations, shared library
libopenblas-dev:arm64 0.2.20+ds-4 arm64 Optimized BLAS (linear algebra) library (development files)
```

Otherwise, run the following command to install the software. You can change the following command to install only some of them as required.

```bash
apt-get install zlib1g-dev libsqlite3-dev libssl-dev libffi-dev unzip pciutils net-tools libblas-dev gfortran
```
CAUTION

- If you install dependencies as the root user, delete sudo from the commands in steps 1 and 2.
- If Python and its dependency are installed as a non-root user, run the su - username command to switch to the non-root user, and then perform step 1 to step 3.
- libsqlite3-dev must be installed before the Python installation. If the Python 3.7.5 environment has been installed in the user's OS and libsqlite3-dev is installed after the Python 3.7.5 environment is installed, you need to rebuild the Python environment.

```
sudo apt-get install -y gcc g++ make cmake zlib1g zlib1g-dev openssl libsqlite3-dev libssl-dev libffi-dev unzip pcreutils net-tools libblas-dev gfortran libblas3 libopenblas-dev libncursesw5-dev
```

**Step 2** Check whether the Python development environment that meets the version requirements is installed in the system.

Run the `python3.7 --version` command. If the returned information meets the Python version requirements (3.7.0 to 3.7.9), go to the next step.

Otherwise, use the following procedure to install Python 3.7.5:

1. Run the following command to download the source code package of Python 3.7.5 to any directory of the installation environment.
   `wget https://www.python.org/ftp/python/3.7.5/Python-3.7.5.tgz`

2. Run the following command to go to the download directory and decompress the source code package:
   `tar -zxvf Python-3.7.5.tgz`

3. Go to the decompressed folder and run the following configuration, build, and installation commands:
   `cd Python-3.7.5`
   `./configure --prefix=/usr/local/python3.7.5 --enable-loadable-sqlite-extensions --enable-shared`
   `make`
   `sudo make install`

   The `--prefix` parameter specifies the Python installation path. You can change it based on the site requirements. The `--enable-shared` parameter is used to compile the `libpython3.7m.so.1.0` dynamic library. The `--enable-loadable-sqlite-extensions` parameter is used to load `libsqlite3-dev`.

   This document uses `--prefix=/usr/local/python3.7.5` as an example. After the configuration, compilation, and installation commands are executed, the installation package is stored in the `/usr/local/python3.7.5` directory, and the `libpython3.7m.so.1.0` dynamic library is stored in the `/usr/local/python3.7.5/lib/libpython3.7m.so.1.0` directory.

4. Run the following commands to set the soft link:
   `sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/local/python3.7.5/bin/python3.7.5`
   `sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/local/python3.7.5/bin/pip3.7.5`

5. Set the Python 3.7.5 environment variables.
   ```
   # Set the Python 3.7.5 library path.
   export LD_LIBRARY_PATH=/usr/local/python3.7.5/lib
   # If multiple Python 3 versions exist in the user environment, use Python 3.7.5.
   export PATH=/usr/local/python3.7.5/bin:
   ```
NOTE

To ensure that the python3.7.5 environment variables can be automatically configured when you install the CANN software package and run related services, create the use_private_python.info file in advance. The following is an example:

1. Run the following command to create the file:

   - User root:
     
     vi /etc/use_private_python.info

   - Non-root user. The HwHiAiUser user is used as an example.
     
     vi /home/HwHiAiUser/use_private_python.info

2. Add the following content to the file:

   
   python37_install_path=/usr/local/python3.7.5

   In the preceding command, /usr/local/python3.7.5 indicates the Python 3.7.5 installation path. Replace it as required.

3. Run the :wq! command to save the file and exit.

6. After the installation is complete, run the following commands to check the installation version. If the required version information is displayed, the installation is successful.

   python3.7 --version
   pip3.7 --version

Step 3  Before the installation, run the pip3.7 list command to check whether the dependencies have been installed. If yes, skip this step. If not, run the following command to install the dependencies. If only some of the software is not installed, modify the following command to install selected software only.

   - Configure the pip source before the installation. For details, see 10.5 Configuring the PIP Source.
   - Run the pip3.7 --version command to query the version. If the version is earlier than 18.0, run the pip3.7 install --upgrade pip command to perform an upgrade.
   - If you run the following commands as a non-root user, add --user at the end of each installation command, for example, pip3.7 install attrs --user. The installation command can be run in any path.
   - The NumPy version must be 1.13.3 or later and earlier than 1.20. The recommended NumPy version is 1.17.2.

   pip3.7 install attrs
   pip3.7 install numpy==1.17.2
   pip3.7 install decorator
   pip3.7 install sympy
   pip3.7 install cffi
   pip3.7 install pyyaml
   pip3.7 install pathlib2
   pip3.7 install psutil
   pip3.7 install protobuf
   pip3.7 install scipy
   pip3.7 install requests

If the error message "subprocess.CalledProcessError: Command 'lsb_release', '-a')' return non-zero exit status 1" is displayed, rectify the fault by referring to 12.7 Error Message "subprocess.CalledProcessError: Command 'lsb_release', '-a')' return non-zero exit status 1" Is Displayed During pip3.7 Installation.

----End
### 5.3.4 Installation Procedure (CentOS 7.6)

#### Checking the Source Validity

The installation requires the download of related dependencies. Ensure that the installation environment can be connected to the network.

Run the following command as the *root* user to check whether the source is valid:

```bash
yum makecache
```

If an error is reported during command execution or dependency installation, check whether the network is connected, or replace the source in the `/etc/yum.repos.d/xxxx.repo` file with an available source or use an image source. For details about how to configure a network proxy, see 10.3 Configuring a System Network Proxy.

**NOTE**

If the message "Your license is invalid" is displayed after you run the preceding command, obtain the OS license.

#### Configuring Permissions for the Installation User

Skip the following part if you install as the *root* user.

Perform the following steps as the *root* user:

1. Run the following command to install sudo:
   ```bash
   yum install sudo
   ```

2. Open the `/etc/sudoers` file:
   ```bash
   chmod u+w /etc/sudoers
   vi /etc/sudoers
   ```

3. Add the following content under `# User privilege specification` in the file:
   ```bash
   username ALL=(ALL:ALL) NOPASSWD:SETENV:/usr/bin/yum, /usr/bin/pip, /bin/tar, /bin/mkdir, /bin/sh, /bin/bash, /bin/make install, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/local/python3.7.5/bin/python3.7.5, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/local/python3.7.5/bin/pip3.7.5
   ```

   In the preceding command, `username` indicates the installation user. Change it as required.

   **CAUTION**
   
   - Ensure that the `/etc/sudoers` file contains `#includedir /etc/sudoers.d`. If the file does not contain `#includedir /etc/sudoers.d`, manually add it to the end of the file.
   
   - After the CANN software package has been installed, you can cancel the sudo permission.
   
   - When uninstalling or upgrading the CANN software package, you also need to configure the preceding user permissions.

4. Run `:wq!` to save the file.

5. Run the following command to revoke the write permission on the `/etc/sudoers` file:
chmod u-w /etc/sudoers

Configuring the Maximum Number of Threads

1. Log in to the installation environment as the root user.
2. Configure environment variables to change the maximum number of threads. In the /etc/profile file, add the following content to the end of the file, save the file, and exit:
   ulimit -u unlimited
3. Run the following command to make the environment variable take effect:
   source /etc/profile

Installing Dependencies

Step 1  Check whether the Python dependencies and GCC software are installed.

Run the following commands to check whether the dependencies such as GCC, Make, and Python are installed:

gcc --version

g++ --version

make --version

cmake --version

rpm -qa | grep unzip

rpm -qa | grep zlib-devel

rpm -qa | grep libffi-devel

rpm -qa | grep openssl-devel

rpm -qa | grep pcap-devel

rpm -qa | grep pcre-devel

rpm -qa | grep gcc-gfortran

If the following information is displayed, the dependencies have been installed. Go to the next step. (The following information shows an example.)

gcc (GCC) 4.8.5 20150623 (Red Hat 4.8.5-39)
g++ (GCC) 4.8.5 20150623 (Red Hat 4.8.5-39)
GNU Make 3.82

cmake version 2.8.12.2

unzip-6.0-21.el7.aarch64

zlib-devel-1.2.7-18.el7.aarch64

libffi-devel-3.0.13-18.el7.aarch64

openssl-devel-1.0.2k-19.el7.aarch64

pcouits-3.5.1-3.el7.aarch64

net-tools-2.0-0.25.20131004git.el7.aarch64

sqlite-devel-3.7.17-8.el7_7.1.aarch64

lapack-devel-3.4.2-8.el7.aarch64

openblas-devel-0.3.3-2.el7.aarch64

gcc-gfortran-4.8.5-39.el7.aarch64

Otherwise, run the following command to install the software. You can change the following command to install only some of them as required.
**CAUTION**

- If you install dependencies as the **root** user, delete `sudo` from the commands in step 1 to step 3.
- If Python and its dependency are installed as a non-root user, run the `su - username` command to switch to the non-root user, and then perform step 1 to step 3.
- SQLite-devel must be installed before the Python installation. If the Python 3.7.5 environment has been installed in the user's OS and SQLite-devel is installed after the Python 3.7.5 environment is installed, you need to rebuild the Python environment.

```bash
sudo yum install -y gcc gcc-c++ make cmake unzip zlib-devel libffi-devel openssl-devel pcre pcreutils net-tools
sqlite-devel lapack-devel openblas-devel gcc-gfortran
```

In the preceding steps, the message "No package libopenblas available" is displayed during the installation of openblas-devel. Run the following command to install the Linux extension package of the enterprise edition:

```bash
sudo yum install epel-release
```

If the CMake version installed using the preceding method is earlier than 3.5.1, see **10.4 Installing CMake 3.5.2**.

**Step 2** Check whether the Python development environment that meets the version requirements is installed in the system.

Run the `python3.7 --version` command. If the returned information meets the Python version requirements (3.7.0 to 3.7.9), go to the next step.

Otherwise, use the following procedure to install Python 3.7.5:

1. Run the following command to download the source code package of Python 3.7.5 to any directory of the installation environment.

   ```bash
   wget https://www.python.org/ftp/python/3.7.5/Python-3.7.5.tgz
   ```

2. Run the following command to go to the download directory and decompress the source code package:

   ```bash
   tar -zxvf Python-3.7.5.tgz
   ```

3. Go to the decompressed folder, create an installation directory, and run the configuration, build, and installation commands.

   ```bash
   cd Python-3.7.5
   ./configure --prefix=/usr/local/python3.7.5 --enable-loadable-sqlite-extensions --enable-shared
   make
   sudo make install
   ```

   - `--prefix` specifies the Python installation path. You can modify it as required.
   - `--enable-shared` is used to build the `libpython3.7m.so.1.0` dynamic library. `--enable-loadable-sqlite-extensions` is used to load the sqlite-devel dependency.

   This document uses `--prefix=/usr/local/python3.7.5` as an example. After the configuration, compilation, and installation commands are executed, the installation package is stored in the `/usr/local/python3.7.5` directory, and the `libpython3.7m.so.1.0` dynamic library is stored in the `/usr/local/python3.7.5/lib/libpython3.7m.so.1.0` directory.

4. Run the following commands to set the soft link:

   ```bash
   sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/local/python3.7.5/bin/python3.7.5
   sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/local/python3.7.5/bin/pip3.7.5
   ```
5. Set the Python 3.7.5 environment variables.

### Set the Python 3.7.5 library path.

```
export LD_LIBRARY_PATH=/usr/local/python3.7.5/lib:$LD_LIBRARY_PATH
```

### If multiple Python 3 versions exist in the user environment, use Python 3.7.5.

```
export PATH=/usr/local/python3.7.5/bin:$PATH
```

**NOTE**

To ensure that the python3.7.5 environment variables can be automatically configured when you install the CANN software package and run related services, create the `use_private_python.info` file in advance. The following is an example:

1. Run the following command to create the file:
   - **User root**:
     ```
     vi /etc/use_private_python.info
     ```
   - **Non-root user**. The `HwHiAiUser` user is used as an example.
     ```
     vi /home/HwHiAiUser/use_private_python.info
     ```

2. Add the following content to the file:
   ```
   python37_install_path=/usr/local/python3.7.5
   ```
   In the preceding command, `/usr/local/python3.7.5` indicates the Python 3.7.5 installation path. Replace it as required.

3. Run the `.wq!` command to save the file and exit.

6. After the installation is complete, run the following commands to check the installation version. If the required version information is displayed, the installation is successful.

   ```
   python3.7 --version
   pip3.7 --version
   ```

### Step 3
Before the installation, run the `pip3.7 list` command to check whether the dependencies have been installed. If yes, skip this step. If not, run the following command to install the dependencies. If only some of the software is not installed, modify the following command to install selected software only.

- **Configure the pip source before the installation. For details, see 10.5 Configuring the PIP Source.**
- Run the `pip3.7 --version` command to query the version. If the version is earlier than 18.0, run the `pip3.7 install --upgrade pip` command to perform an upgrade.
- If you run the following commands as a non-root user, add `--user` at the end of each installation command, for example, `pip3.7 install attrs --user`. The installation command can be run in any path.
- The NumPy version must be 1.13.3 or later and earlier than 1.20. The recommended NumPy version is 1.17.2.

```
pip3.7 install attrs
pip3.7 install numpy==1.17.2
pip3.7 install decorator
pip3.7 install sympy
pip3.7 install cffi
pip3.7 install pyyaml
pip3.7 install pathlib2
pip3.7 install psutil
pip3.7 install protobuf
pip3.7 install scipy
pip3.7 install requests
```

- If an error is reported during the NumPy installation, rectify the fault by referring to 12.6 pip3.7 install numpy Error.
• If an error is reported during the SciPy installation, rectify the fault by referring to 12.5 pip3.7 install scipy Error.

**Step 4** Skip this step if only offline inference is required.

CentOS 7.6 uses GCC 4.8.5 by default. For details about how to install GCC 7.3.0, see 10.6 Installing GCC 7.3.0.

----End

# 5.4 Installing Software Packages on a Non-Ascend Device

## 5.4.1 Installing the Development Kit

**Prerequisites**

- The preparations for the installation are complete. For details, see 5.3 Installing OS Dependencies.
- The development kit (Ascend-cann-toolkit_xxx.run) for two architectures has been obtained. For details, see 5.1 Obtaining Software Packages.

**Procedure**

**Step 1** Log in to the installation environment as the installation user of the software package.

If the dependency installation user in 5.3 Installing OS Dependencies is the root user, you can specify an installation user of the software package. If the dependency installation user in 5.3 Installing OS Dependencies is a non-root user, ensure that the installation user of the software package is the same as the dependency installation user.

**Step 2** Upload the obtained development kit to any directory (for example, /home/package) in the installation environment.

**Step 3** Go to the directory where the software package is stored.

**Step 4** Grant the execute permission on the software package.

    chmod +x *.run

**NOTE**

In the preceding command, *.run indicates the development kit Ascend-cann-toolkit_{version}_linux-{arch}.run. Replace it with the actual package name.

**Step 5** Run the following command to check the consistency and integrity of the software package installation file:

    ./*.run --check

**Step 6** Install the software. (The following commands support --install-for-all and --install-path=<path>. For details about the parameters, see 13.1 Parameters.)
If the installation is performed as a non-root user, run the following command:

```
./*.run --install
```

If the installation is performed as the root user:

- As the default running user `HwHiAiUser`, run the following command:
  
  ```
  ./*.run --install
  ```

- As a specified running user, run the following command:
  
  ```
  ./*.run --install-username=username --install-usergroup=usergroup --install
  ```

  where, `--install-username` and `--install-usergroup` are used to specify the running users.

**NOTE**

- If the installation is performed by the root user, **do not specify the installation path in the directory of a non-root user**. Otherwise, the root user file may be replaced by a non-root user for privilege escalation.

- If you do not specify an installation path, the software is installed in the default path. The default installation paths are as follows:
  - root user: `/usr/local/Ascend`
  - Non-root user: `${HOME}/Ascend`

  In the preceding command, `${HOME}` indicates the directory of the current user.

If the following information is displayed, the software has been installed:

```
[INFO] xxx install success
```

`xxx` indicates the name of the software package to be installed.

**5.4.2 Configuring the Cross Compilation Environment**

For the Atlas 200 AI accelerator module (RC scenario) and Atlas 500 AI edge station, prepare an x86 server (or PC) for setting up the development environment. In this case, you need to install the cross compilation tool in the development environment. For details, see Table 5-4.
Table 5-4 Cross-compiler installation

<table>
<thead>
<tr>
<th>Development Environment Architecture</th>
<th>Operating Environment Architecture</th>
<th>Compilation Environment Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>x86_64</td>
<td>aarch64</td>
<td>Run the \texttt{aarch64-linux-gnu-g++ --version} command in the development environment as the installation user of the software package to check whether the software package has been installed. If the software package has been installed, skip this step. An example of the installation command is as follows. Run the command based on the actual situation. \texttt{sudo apt-get install g++-aarch64-linux-gnu}</td>
</tr>
</tbody>
</table>

5.4.3 Follow-up

Developers can develop applications in the development environment. For details, see the \textit{CANN Application Software Development Guide (C and C++)} or \textit{CANN Application Software Development Guide (Python)}.

5.5 Installing Software Packages on an Ascend Device

5.5.1 Installing the Development Kit

Prerequisites

- The preparations for the installation are complete. For details, see 5.3 Installing OS Dependencies.
- The development kit package \texttt{Ascend-cann-toolkit\_xxx.run} has been obtained. For details, 5.1 Obtaining Software Packages.

Procedure

\textbf{Step 1} Log in to the installation environment as the installation user of the software package.

If the dependency installation user in 5.3 Installing OS Dependencies is the root user, you can specify an installation user of the software package. If the dependency installation user in 5.3 Installing OS Dependencies is a non-root user, ensure that the installation user of the software package is the same as the dependency installation user.

\textbf{Step 2} Upload the obtained development kit to any directory (for example, \texttt{/home/package}) in the installation environment.
**Step 3** Go to the directory where the software package is stored.

**Step 4** Grant the execute permission on the software package.

```bash
chmod +x *.run
```

**NOTE**

In the preceding command, `*.run` indicates the development kit `Ascend-cann-toolkit_{version}_linux-{arch}.run`. Replace it with the actual package name.

**Step 5** Run the following command to check the consistency and integrity of the software package installation file:

```bash
./*.run --check
```

**Step 6** Install the software. (The following commands support `--install-for-all` and `--install-path=<path>`. For details about the parameters, see 13.1 Parameters.)

- If the installation is performed as a non-root user, run the following command:
  ```bash
  ./*.run --install
  ```

- If the installation is performed as the root user:
  - As the default running user `HwHiAiUser`, run the following command:
    ```bash
    ./*.run --install
    ```
  - As a specified running user, run the following command:
    ```bash
    ./*.run --install-username=username --install-usergroup=usergroup --install
    ```
    where, `--install-username` and `--install-usergroup` are used to specify the running users.

**NOTE**

- The development kits of multiple versions can be installed by different users in the same development environment. The users must be in the same group as the driver running user. If the owner groups are different, add the users to the group of the driver running user.

- If the installation is performed by the root user, do not to specify the installation path in the directory of a non-root user. Otherwise, the root user file may be replaced by a non-root user for privilege escalation.

- If you do not specify an installation path, the software is installed in the default path. The default installation paths are as follows:

  - root user: `/usr/local/Ascend`
  - Non-root user: `$HOME/Ascend`

  In the preceding command, `$HOME` indicates the directory of the current user.

If the following information is displayed, the software is successfully installed:

```
[INFO] xxx install success
```

`xxx` indicates the name of the software package to be installed.

----End

### 5.5.2 Installing the Framework Plugin Package

If the deep learning framework TensorFlow is used for online inference or training, install the framework plugin package `Ascend-cann-tfplugin_xxx.run`. 
Install the framework plugin package by referring to 5.5.1 Installing the Development Kit.

5.5.3 Installing the Deep Learning Framework

If you perform only offline inference, skip this section.

This document uses TensorFlow as an example to describe how to install the deep learning framework. If you want to use the MindSpore framework, visit https://www.mindspore.cn/install/en to obtain the method of installing the MindSpore framework. If you want to use the PyTorch framework, see "Installing the PyTorch Framework" in the PyTorch Network Model Porting and Training Guide.

Installation Preparations

- For the x86 architecture, skip the installation preparations.
- For the AArch64 architecture:
  - TensorFlow depends on h5py, and h5py depends on HDF5. Therefore, you need to compile and install HDF5 first. Otherwise, an error is reported when you use pip to install h5py. Perform the following operations as the root user.
  - TensorFlow depends on grpcio. Therefore, you are advised to install the grpcio of the specified version before installing TensorFlow. Otherwise, the installation may fail.

Step 1 Compile and install the HDF5.

1. Run the following command to download the source code package of HDF5 to any directory of the installation environment:
   ```bash
   wget https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.5/src/hdf5-1.10.5.tar.gz --no-check-certificate
   ```
2. Run the following command to go to the download directory and decompress the source code package:
   ```bash
   tar -zxvf hdf5-1.10.5.tar.gz
   ```
   Go to the decompressed folder and run the following configuration, build, and installation commands:
   ```bash
   cd hdf5-1.10.5/
   ./configure --prefix=/usr/include/hdf5
   make
   make install
   ```

Step 2 Configure environment variables and create a soft link to the dynamic link library (DLL).

1. Configure environment variables:
   ```bash
   export CPATH="/usr/include/hdf5/include:/usr/include/hdf5/lib/"
   ```
2. Run the following commands as the root user to create a soft link to the DLL. Add sudo before the following commands as a non-root user:
   ```bash
   ln -s /usr/include/hdf5/lib/libhdf5.so /usr/lib/libhdf5.so
   ln -s /usr/include/hdf5/lib/libhdf5_hl.so /usr/lib/libhdf5_hl.so
   ```

Step 3 Install h5py.

Run the following command as the root user to install the h5py dependency:
```bash
pip3.7 install Cython
```

The h5py installation command is as follows:
pip3.7 install h5py==2.8.0

Step 4  Install grpcio.

Run the following command as the root user to install grpcio:

pip3.7 install grpcio==1.32.0

----End

Installing TensorFlow 1.15.0

TensorFlow 1.15 is necessary to develop and verify operators and develop training services.

- For the x86 architecture, download the software package from the pip source. For details, see https://www.tensorflow.org/install/pip?lang=python3. Note that the instructions provided by the TensorFlow website are incorrect. To download the CPU version from the pip source, you need to explicitly specify tensorflow-cpu. Otherwise, the GPU version is downloaded by default. That is, change tensorflow==1.15 —Release for CPU-only to tensorflow-cpu==1.15 —Release for CPU-only. In addition, the installation command pip3 install --user --upgrade tensorflow described on the official website needs to be changed to pip3.7 install tensorflow-cpu==1.15 as the root user and to pip3.7 install tensorflow-cpu==1.15 --user as a non-root user.

- For the AArch64 architecture, perform the following operations:
  a. Download the compiled TensorFlow 1.15 software package:
     tensorflow-1.15.0-cp37-cp37m-linux_aarch64.whl
  b. Install the compiled TensorFlow.
     Go to the directory where the software package is stored and run the following command as the root user to install TensorFlow 1.15:
     pip3.7 install tensorflow-1.15.0-cp37-cp37m-linux_aarch64.whl
     Run the following command as a non-root user:
     pip3.7 install tensorflow-1.15.0-cp37-cp37m-linux_aarch64.whl --user

5.5.4 Installing protobuf Python

If the training script depends on the Python version of protobuf to store data in the serialized structure (for example, the serialization interfaces of TensorFlow), you need to install protobuf Python.

Step 1  Check whether the system contains the dynamic library /usr/local/python3.7.5/lib/python3.7/site-packages/google/protobuf/pyext/_message.cpython-37m-<arch>-linux-gnu.so (Python 3.7.5 installed by referring to 5.3 Installing OS Dependencies is used as an example). If not, perform the following steps to install it: <arch> indicates the system architecture type.

- NOTE

/usr/local/python3.7.5/lib/python3.7/site-packages is the path for installing third-party libraries using pip. You can run the pip3.7 -V command to query the path.
If the system displays /usr/local/python3.7.5/lib/python3.7/site-packages/pip, the path for installing third-party libraries using pip is /usr/local/python3.7.5/lib/python3.7/site-packages.
Step 2  Uninstall protobuf.

```
python3.7 uninstall protobuf
```

Step 3  Download the protobuf software package.

Download the protobuf-python-3.11.3.tar.gz software package (or another version that is compatible with TensorFlow installed in the current environment) from https://github.com/protocolbuffers/protobuf/releases/download/v3.11.3/protobuf-python-3.11.3.tar.gz, upload the package to any directory on the Linux server as the root user, and run the `tar zxf protobuf-python-3.11.3.tar.gz` command to decompress the package.

Step 4  Install protobuf as the root user.

Go to the protobuf software package directory.

1. Install the dependencies of protobuf.
   
   If the OS is Ubuntu, run the following command:
   ```
apt-get install autoconf automake libtool curl make g++ unzip libffi-dev -y
   ```
   
   If the OS is EulerOS, run the following command:
   ```
yum install autoconf automake libtool curl make gcc-c++ unzip libffi-devel -y
   ```
   
   If the OS is CentOS or BCLinux, run the following command:
   ```
yum install autoconf automake libtool curl make gcc-c++ unzip libffi-devel -y
   ```
   
2. Grant the execute permission on the autogen.sh script and execute the script.
   ```
   chmod +x autogen.sh
   ./autogen.sh
   ```
   
3. Configure the installation path. The default installation path is `/usr/local`
   ```
   ./configure
   ```
   
   To specify the installation path, run the following command:
   ```
   ./configure --prefix=/protobuf
   ```
   
   /protobuf indicates the installation path specified by the user.
   
4. Run the following commands to install protobuf:
   ```
   make -j15  # Check the number of CPUs by running grep -w processor /proc/cpuinfo|wc -l. In this example, the number is 15. You can set the parameters as required.
   make install
   ```
   
5. Refresh the shared library.
   ```
   ldconfig
   ```
   
   After protobuf is installed, a `google/protobuf` folder is generated in the include directory under the path configured in Step 4.3 to store the related header files of protobuf. In addition, a .protoc executable file is generated in the bin directory under the path configured in Step 4.3. The .protoc executable file is used to build the .proto file and generate the C++ header file and implementation file of protobuf.
   
6. Check whether the software package is successfully installed.
   ```
   ln -s /protobuf/bin/protoc /usr/bin/protoc
   protoc --version
   ```
/protobuf is the installation path configured in Step 4.3. If the installation path is not configured, run the protoc --version command to check whether the installation is successful.

Step 5  Install the runtime library of protobuf Python.

1. Go to the python subdirectory in the protobuf software package directory and compile the Python runtime library.

   python3.7 setup.py build --cpp_implementation

   **NOTE**
   Build a runtime library of the binary version. If the runtime library of the binary version cannot be generated by running the python3.7 setup.py build command, the processing of the serialization structure is slow.

2. Install the dynamic library.

   cd .. && make install

   Go to the python subdirectory and install the Python runtime library.

   python3.7 setup.py install --cpp_implementation

3. Check whether the library has been installed.

   Check whether the dynamic library /usr/local/python3.7.5/lib/python3.7/site-packages/protobuf-3.11.3-py3.7-linux-aarch64.egg/google/protobuf/pyext/_message.cpython-37m-<arch>-linux-gnu.so exists in the system. <arch> indicates the system architecture type.

   **NOTE**
   /usr/local/python3.7.5/lib/python3.7/site-packages is the path for installing third-party libraries using pip. You can run the pip3.7 -V command to query the path.

   If the system displays /usr/local/python3.7.5/lib/python3.7/site-packages/pip, the path for installing third-party libraries using pip is /usr/local/python3.7.5/lib/python3.7/site-packages.

4. If you have specified the installation path in Step 4.3, add the following environment variable settings to the run script:

   export LD_LIBRARY_PATH=/protobuf/lib:$LD_LIBRARY_PATH

   In the preceding command, /protobuf indicates the installation path set by the user in Step 4.3.

5. Establish a soft link.

   If you configure the installation path, you need to establish a soft link. Otherwise, an error will be reported when TensorFlow is imported. The command is as follows:

   ln -s /protobuf/lib/libprotobuf.so.22.0.3 /usr/lib/libprotobuf.so.22

   /protobuf is the installation path configured in Step 4.3.

---End

5.5.5 Configuring the NIC IP Address of a Device

During distributed training, you need to use the HCCN Tool in the Ascend software to configure the NIC IP address of the device. This IP address is used for communication between devices to synchronize network model parameters. This section describes only the commands for configuring the network by using the
HCCN tool. If you need to use other functions of the HCCN tool (for example, checking the link status of a network port), see the Ascend 910 HCCN Tool Interface Reference.

Atlas 800 training server and Atlas 900 AI cluster

**NOTE**

To check whether the SMP or AMP mode is used, log in to the BMC background and run the `ipmcget -d npuworkmode` command.

- **SMP** (symmetric multi-processor) mode

  Log in to the AI Servers as the **root** user and configure the NIC IP address of each device. The configuration requirements are as follows:
  
  - NICs 0 and 4, 1 and 5, 2 and 6, and 3 and 7 of an AI Server must be in the same network segment respectively. NICs 0, 1, 2, and 3 must be in different network segments. NICs 4, 5, 6, and 7 must be in different network segments.
  
  - In the cluster scenario, the devices in the similar positions on AI Servers must be in the same network segment. For example, NIC 0 of AI Server 1 and AI Server 2 must be in the same network segment, and NIC 1 of AI Server 1 and AI Server 2 must be in the same network segment. Change the IP address as required.

  ```
  hccn_tool -i 0 -ip -s address 192.168.100.101 netmask 255.255.255.0
  hccn_tool -i 1 -ip -s address 192.168.101.101 netmask 255.255.255.0
  hccn_tool -i 2 -ip -s address 192.168.102.101 netmask 255.255.255.0
  hccn_tool -i 3 -ip -s address 192.168.103.101 netmask 255.255.255.0
  hccn_tool -i 4 -ip -s address 192.168.100.100 netmask 255.255.255.0
  hccn_tool -i 5 -ip -s address 192.168.101.100 netmask 255.255.255.0
  hccn_tool -i 6 -ip -s address 192.168.102.100 netmask 255.255.255.0
  hccn_tool -i 7 -ip -s address 192.168.103.100 netmask 255.255.255.0
  ```

- **AMP** (asymmetric multi-processor) mode

  In AMP mode, you do not need to configure the NIC IP address of the device.

**Atlas 300T training card**

Each server can be configured with one Atlas 300T training card or two. Each card corresponds to one Device OS and needs to be configured with one IP address. Different cards need to be configured with IP addresses in the same network segment.

Log in to the AI Servers as the **root** user and configure the NIC IP address of each device. The configuration operations are as follows:

**Step 1** Run the `npu-smi info` command to view the ID of the device to be configured. In Figure 5-1, the NPU IDs are 1 and 4, for example. Use the actual NPU IDs in the query result.

**Figure 5-1 Checking the device ID**
Step 2 Run the following commands to configure the NIC IP addresses of the device. The IP addresses used in the following example are for reference only.

```
hccn_tool -i 1 -ip -s address 192.168.0.2 netmask 255.255.255.0
hccn_tool -i 4 -ip -s address 192.168.0.3 netmask 255.255.255.0
```

----End

**NOTE**
- Ensure that the npu-smi tool has been installed on the server.
- In cluster scenarios, the devices of each AI server must be in the same network segment.

### 5.5.6 Configuring Environment Variables

The CANN software provides a script for setting process-level environment variables. You can reference the script in processes to automatically configure environment variables. The environment variables automatically become invalid after the processes end. The default installation path of the root user is used as an example.

```
# Configure environment variables when installing the Toolkit package.
. /usr/local/Ascend/ascend-toolkit/set_env.sh

# Configure environment variables when installing the tfplugin package.
. /usr/local/Ascend/tfplugin/set_env.sh
```

You can also configure permanent environment variables by modifying the `~/.bashrc` file. The procedure is as follows:

1. Run the `vi ~/.bashrc` command in any directory as the running user to open the `.bashrc` file and append the preceding lines to the file.
2. Run the `:wq!` command to save the file and exit.
3. Run the `source ~/.bashrc` command for the modification to take effect immediately.

### 5.5.7 Follow-up Procedure

- Develop inference applications in the development environment. For details, see the CANN Application Software Development Guide (C and C++) or CANN Application Software Development Guide (Python).
- Port models. For details, see the CANN TensorFlow Network Model Porting and Training Guide or PyTorch Network Model Porting and Training Guide.
- Obtain an offline model or training script from ModelZoo.
6 Installing the Operating Environment (Deploying the nnrt Software)

6.1 Before You Start

6.2 Obtaining Software Packages

6.3 Installing the Operating Environment on the PM or VM

6.1 Before You Start

This section describes how to install an operating environment that supports offline inference.

The operating environment is an environment for running applications developed by users.

- For details about how to install the operating environment of the Atlas 200 AI accelerator module (RC scenario), see the Atlas 200 AI Accelerator Module 1.0.10 Software Installation and Maintenance Guide (RC, Model 3000).
- An Atlas 500 AI edge station is preinstalled with the EulerOS, driver, and firmware. For details about the initial configuration of the Atlas 500 AI edge station, see "Installation and Configuration > Initial Configuration" in the Atlas 500 AI Edge Station User Guide (Models 3000, 3010).
- Figure 6-1 shows the software to be installed for the operating environment on other Ascend devices (only offline inference is supported). You can deploy physical machines or VMs.
  - If you need to deploy a physical machine or VM, install the operating environment by referring to 6.3 Installing the Operating Environment on the PM or VM.
6.2 Obtaining Software Packages

Downloading Software Packages

Obtain required software packages based on the following table. The versions of the software packages must be consistent.

Table 6-1 CANN software packages

<table>
<thead>
<tr>
<th>Package Type</th>
<th>Package Name</th>
<th>Description</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offline inference engine package</td>
<td>Ascend-cann-nnrt_{version}_linux-{arch}.run</td>
<td>It supports offline inference only and is used for model inference of applications. This package includes AscendCL libraries and building dependencies (excluding the libraries in the driver package).</td>
<td>Link</td>
</tr>
</tbody>
</table>

NOTE

_{version}_ indicates the software version, and _{arch}_ indicates the CPU architecture.

Checking the Integrity of Software Packages

To prevent a software package from being maliciously tampered with during transmission or storage, download the corresponding verification file for integrity verification when downloading the software package.

If the extension of the downloaded verification file is .sha256sum, perform the following steps:

Place the downloaded software package and its verification file in the same directory and run the following command as the root user:

```
sha256sum -c Ascend-cann-nnrt_{software version}_linux-{arch}.run.sha256sum
```
6.3 Installing the Operating Environment on the PM or VM

6.3.1 Preparing Installation and Running Users

- Running user: user who runs the inference service or performs training.
- Installation user: user who installs the software package.
  - If the root user is used for installation, you are advised to use a non-root user to run related services. Before the installation, create a running user.
  - If a non-root user is used for installation, the installation and running users must be the same.
- For installation as the root user:
  - (Recommended) If the created running user is HwHiAiUser, which is the default running user of the CANN software package, you do not need to specify the running user during installation.
  - If the created running user is not HwHiAiUser, you need to specify the running user when installing the CANN software package.
- For installation as a non-root user:
  - If a non-root user exists, you do not need to create one.
  - To create a non-root user, perform the following steps:

## NOTICE

- If the CANN software package is installed by a non-root user, ensure that the owner group of the non-root user is the same as that of the user who runs the driver. If they are different, add the non-root user to the group of the driver running user.
- The created running user cannot belong to the root user group because security risks may exist.

Run the following commands as the root user to create a non-root user.

1. Create a non-root user.

```bash
groupadd usergroup
useradd -g usergroup -d /home/username -m username -s /bin/bash
```

Create the running user HwHiAiUser.

```bash
groupadd HwHiAiUser
useradd -g HwHiAiUser -d /home/HwHiAiUser -m HwHiAiUser -s /bin/bash
```

2. Set the password for the non-root user.

```bash
passwd username
```

The following is an example:

```bash
passwd HwHiAiUser
```
After the HwHiAiUser user is created, do not disable the login authentication function of the user.

The password validity period is 90 days. You can change the validity period in the `/etc/login.defs` file or using the `chage` command. For details, see 10.8 Setting User Account Validity Period.

## 6.3.2 Installing the Offline Inference Engine Package

### Prerequisites

- The preparations for the installation are complete. For details, see 6.3.1 Preparing Installation and Running Users.
- The inference card or AI accelerator module driver and firmware have been installed in the installation environment.
- The offline inference engine package nnrt has been obtained. For details, see 6.2 Obtaining Software Packages.

### Procedure

To obtain and install the offline inference engine package, perform the following steps:

**Step 1** Log in to the installation environment as the installation user of the software package.

**Step 2** Upload the offline inference engine package obtained to any path (for example, `/home/package`) in the installation environment.

**Step 3** Go to the directory where the software package is stored.

**Step 4** Grant the execute permission on the software package.

```
chmod +x *.run
```

*NOTE*

`.run` indicates the software package name, for example, the offline inference engine package `Ascend-cann-nnrt_{version}_linux-{arch}.run`. Replace it with the actual package name.

**Step 5** Run the following command to check the consistency and integrity of the software package installation file:

```
./*.run --check
```

**Step 6** Install the software. (The following commands support `--install-for-all` and `--install-path=<path>`.) For details about the parameters, see 13.1 Parameters.

- If the installation is performed as a non-root user, run the following command:
  
  `./*.run --install`

- If the installation is performed as the root user:
  
  - As the default running user `HwHiAiUser`, run the following command:
    
    `./*.run --install`
As a specified running user, run the following command:

```
./run --install-username=username --install-usergroup=usergroup --install
```

where, `--install-username` and `--install-usergroup` are used to specify the running users.

**NOTE**

- The offline inference engine package can be installed by different users in the same operating environment. However, for different users, the installed versions must be the same, and users must be in the same group as the driver running user. Otherwise, add the users to the group of the driver running user.
- If the installation is performed by the `root` user, do not specify the installation path in the directory of a non-root user. Otherwise, the `root` user file may be replaced by a non-root user for privilege escalation.
- If you do not specify an installation path, the software is installed in the default path. The default installation paths are as follows:
  - `root` user: `/usr/local/Ascend`
  - Non-root user: `${HOME}/Ascend`

In the preceding command, `${HOME}` indicates the directory of the current user.

If the following information is displayed, the software is successfully installed:

```
[INFO] xxx install success
```

`xxx` indicates the name of the software package to be installed.

#### 6.3.3 Configuring Environment Variables

The CANN software provides a script for setting process-level environment variables. You can reference the script in processes to automatically configure environment variables. The environment variables automatically become invalid after the processes end. The default installation path of the `root` user is used as an example.

```
# Configure environment variables when installing the nnrt package.
./usr/local/Ascend/nnrt/set_env.sh
```

You can also configure permanent environment variables by modifying the `~/.bashrc` file. The procedure is as follows:

1. Run the `vi ~/.bashrc` command in any directory as the running user to open the `~/.bashrc` file and append the preceding lines to the file.
2. Run the `:wq!` command to save the file and exit.
3. Run the `source ~/.bashrc` command for the modification to take effect immediately.

#### 6.3.4 Follow-up

Obtain offline models from [ModelZoo](https://model-zoo.huawei.com/) for inference.
Installing the Operating Environment
(Deploying the nnae Software)

7.1 Before You Start
7.2 Obtaining Software Packages
7.3 Installing the Operating Environment on the PM or VM

7.1 Before You Start

This section describes how to install the operating environment that supports offline inference, online inference, and training.

Figure 7-1 shows the software to be installed in the operating environment. You can deploy physical machines or VMs.

- If you need to deploy a physical machine or VM, install the operating environment by referring to 7.3 Installing the Operating Environment on the PM or VM.

Figure 7-1 Operating environment
7.2 Obtaining Software Packages

Downloading Software Packages

Obtain required software packages based on the following table. The versions of the software packages must be consistent.

Table 7-1 CANN software packages

<table>
<thead>
<tr>
<th>Name</th>
<th>Package</th>
<th>Description</th>
<th>How to Obtain</th>
</tr>
</thead>
</table>
| Deep learning engine package  | Ascend-cann-nnae_{version}_linux-{arch}.run | - It supports offline and online inference and training, including the Fwklib and OPP.  
- Obtain the software packages based on the CPU architecture (x86_64 or AArch64). | Link          |
| Framework plugin package      | Ascend-cann-tfplugin_{version}_linux-{arch}.run | Plugin package, an adaptation plugin used to connect to the upper-layer framework TensorFlow. 
If the deep learning framework TensorFlow is used for online inference or training, you need to obtain the software package. | Link          |

NOTE

{version} indicates the software version, and <arch> indicates the CPU architecture.

Checking the Integrity of Software Packages

To prevent a software package from being maliciously tampered with during transmission or storage, download the corresponding verification file for integrity verification when downloading the software package.

If the extension of the downloaded verification file is .sha256sum, perform the following steps:

Place the downloaded software package and its verification file in the same directory and run the following command as the root user:

```bash
sha256sum -c Ascend-cann-nnrt_{software version}_linux-aarch64.run.sha256sum
```

The file passes the integrity verification if you see information similar to the following:
7.3 Installing the Operating Environment on the PM or VM

7.3.1 Preparing Installation and Running Users

Checking the umask of the root User

1. Log in to the installation environment as the root user.
2. Check the umask value of the root user.
   umask
3. If the umask value is not 0022, append umask 0022 to the file and save the file.
   a. Run the following command in any directory to open the .bashrc file.
      vi ~/.bashrc
      Append umask 0022 to the end of the file.
   b. Run the :wq! command to save the file and exit.
   c. Run the source ~/.bashrc command for the modification to take effect immediately.

Creating Installation and Running Users

- Running user: user who runs the inference service or performs training.
- Installation user: user who installs the software package.
  - If the root user is used for installation, you are advised to use a non-root user to run related services. Before the installation, create a running user.
  - If a non-root user is used for installation, the installation and running users must be the same.
- For installation as the root user:
  - (Recommended) If the created running user is HwHiAiUser, which is the default running user of the CANN software package, you do not need to specify the running user during installation.
  - If the created running user is not HwHiAiUser, you need to specify the running user when installing the CANN software package.
- For installation as a non-root user:
  - If a non-root user exists, you do not need to create one.
  - To create a non-root user, perform the following steps:
If the CANN software package is installed by a non-root user, ensure that the owner group of the non-root user is the same as that of the user who runs the driver. If they are different, add the non-root user to the group of the driver running user.

The created running user cannot belong to the root user group because security risks may exist.

Run the following commands as the root user to create a non-root user.

1. Create a non-root user.
   ```bash
   groupadd usergroup
   useradd -g usergroup -d /home/username -m username -s /bin/bash
   ```
   Create the running user HwHiAiUser.
   ```bash
   groupadd HwHiAiUser
   useradd -g HwHiAiUser -d /home/HwHiAiUser -m HwHiAiUser -s /bin/bash
   ```

2. Set the password for the non-root user.
   ```bash
   passwd username
   ```
   Example:
   ```bash
   passwd HwHiAiUser
   ```

**NOTE**

- After the HwHiAiUser user is created, do not disable the login authentication function of the user.
- The password validity period is 90 days. You can change the validity period in the /etc/login.defs file or using the chage command. For details, see 10.8 Setting User Account Validity Period.

### 7.3.2 Installing OS Dependencies

#### 7.3.2.1 Before You Start

Before installing the CANN software, install related dependencies.

This section describes how to install the dependencies on Ubuntu 18.04 and CentOS 7.6. You can refer to the following procedure to install the dependencies on other OSs.

#### 7.3.2.2 Dependency List

- Ubuntu 18.04

<table>
<thead>
<tr>
<th>Type</th>
<th>Version Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>3.7.0–3.7.9</td>
</tr>
<tr>
<td>cmake</td>
<td>≥ 3.5.1</td>
</tr>
<tr>
<td>make</td>
<td>-</td>
</tr>
<tr>
<td>Type</td>
<td>Version Requirement</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| gcc          | ● Offline inference  
The GCC version must be 4.8.5 or later.                                      |
|              | ● Online inference and training  
The GCC version must be 7.3.0 or later. If the GCC version is earlier than 7.3.0, install the GCC by referring to 10.6 Installing GCC 7.3.0. |
| g++          |                                                                                    |
| zlib1g       | There is no version requirement. The version to be installed is subject to the source provided by the OS. |
| zlib1g-dev   |                                                                                    |
| sqlite3-dev  |                                                                                    |
| openssl      |                                                                                    |
| libssl-dev   |                                                                                    |
| libffi-dev   |                                                                                    |
| unzip        |                                                                                    |
| pciutils     |                                                                                    |
| net-tools    |                                                                                    |
| libblas-dev  |                                                                                    |
| gfortran     |                                                                                    |
| libblas3     |                                                                                    |
| liblapack-dev|                                                                                    |
| liblapack3   |                                                                                    |
| libopenblas-dev|                                                                                  |
| numpy        | >=1.13.3 &<1.20                                                                     |
| decorator    | >=4.4.0                                                                             |
| sympy        | >=1.4                                                                               |
| cffi         | >=1.12.3                                                                            |
| protobuf     | >=3.11.3                                                                            |
| attrs        |                                                                                    |
| pyyaml       |                                                                                    |
| pathlib2     |                                                                                    |
| scipy        |                                                                                    |
| requests     |                                                                                    |
| psutil       |                                                                                    |
|              | ● CentOS 7.6                                                                        |
### Table 7-3 Dependency information

<table>
<thead>
<tr>
<th>Type</th>
<th>Version Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>3.7.0–3.7.9</td>
</tr>
<tr>
<td>cmake</td>
<td>≥ 3.5.1</td>
</tr>
<tr>
<td>make</td>
<td>-</td>
</tr>
<tr>
<td>gcc</td>
<td>· Offline inference&lt;br&gt;The GCC version must be 4.8.5 or later.</td>
</tr>
<tr>
<td>gcc-c++</td>
<td>· Online inference and training&lt;br&gt;The GCC version must be 7.3.0 or later. If the GCC version is earlier than 7.3.0, install the GCC by referring to <a href="#">10.6 Installing GCC 7.3.0</a>.</td>
</tr>
<tr>
<td>unzip</td>
<td>There is no version requirement. The version to be installed is subject to the source provided by the OS.</td>
</tr>
<tr>
<td>zlib-devel</td>
<td></td>
</tr>
<tr>
<td>libffi-devel</td>
<td></td>
</tr>
<tr>
<td>openssl-devel</td>
<td></td>
</tr>
<tr>
<td>pciumits</td>
<td></td>
</tr>
<tr>
<td>net-tools</td>
<td></td>
</tr>
<tr>
<td>sqlite-devel</td>
<td></td>
</tr>
<tr>
<td>lapack-devel</td>
<td></td>
</tr>
<tr>
<td>openblas-devel</td>
<td></td>
</tr>
<tr>
<td>gcc-gfortran</td>
<td></td>
</tr>
<tr>
<td>numpy</td>
<td>&gt;=1.13.3 &amp;&amp; &lt;1.20</td>
</tr>
<tr>
<td>decorator</td>
<td>&gt;=4.4.0</td>
</tr>
<tr>
<td>sympy</td>
<td>&gt;=1.4</td>
</tr>
<tr>
<td>cffi</td>
<td>&gt;=1.12.3</td>
</tr>
<tr>
<td>protobuf</td>
<td>&gt;=3.11.3</td>
</tr>
<tr>
<td>attrs</td>
<td></td>
</tr>
<tr>
<td>pyyaml</td>
<td></td>
</tr>
<tr>
<td>pathlib2</td>
<td></td>
</tr>
<tr>
<td>scipy</td>
<td></td>
</tr>
<tr>
<td>requests</td>
<td></td>
</tr>
<tr>
<td>psutil</td>
<td></td>
</tr>
<tr>
<td>There is no version requirement. The version is subject to the pip source.</td>
<td></td>
</tr>
</tbody>
</table>

7 Installing the Operating Environment (Deploying the nnae Software)
7.3.2.3 Installation Procedure (Ubuntu 18.04)

Checking the Source Validity

The installation requires the download of related dependencies. Ensure that the installation environment can be connected to the network.

Run the following command as the root user to check whether the source is valid:

```
apt-get update
```

If an error is reported during command execution or dependency installation, check whether the network is connected, or replace the source in the `/etc/apt/sources.list` file with an available source or use an image source. For details about how to configure a network proxy, see 10.3 Configuring a System Network Proxy.

Configuring Permissions for the Installation User

Skip the following part if you install as the root user.

Perform the following steps as the root user:

1. Run the following command to install sudo:
   ```
   apt-get install sudo
   ```

2. Open the `/etc/sudoers` file:
   ```
   chmod u+w /etc/sudoers
   vi /etc/sudoers
   ```

3. Append the following content to the configuration file:
   ```
   username ALL=(ALL:ALL) NOPASSWD:SETENV:/usr/bin/apt-get, /usr/bin/pip, /bin/tar, /bin/mkdir, /bin/sh, /bin/bash, /usr/bin/make install, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/local/python3.7.5/bin/python3.7.5, /usr/bin/unzip
   ```
   In the preceding command, `username` indicates the installation user. Change it as required.

   **CAUTION**

   - Ensure that the `/etc/sudoers` file contains `#includedir /etc/sudoers.d`. If the file does not contain `#includedir /etc/sudoers.d`, manually add it to the end of the file.
   - After the CANN software package has been installed, you can cancel the sudo permission.
   - When uninstalling or upgrading the CANN software package, you also need to configure the preceding user permissions.

4. Run `:wq!` to save the file.

5. Run the following command to revoke the write permission on the `/etc/sudoers` file:
   ```
   chmod u-w /etc/sudoers
   ```
Installing Dependencies

Step 1  Check whether the Python dependencies and GCC software are installed.

Run the following commands to check whether the dependencies such as GCC, Make, and Python are installed:

```
gcc --version
g++ --version
make --version
cmake --version
dpkg -l zlib1g| grep zlib1g| grep ii
dpkg -l zlib1g-dev| grep zlib1g-dev| grep ii
dpkg -l libsqlite3-dev| grep libsqlite3-dev| grep ii
dpkg -l libssl-dev| grep libssl-dev| grep ii
dpkg -l libffi-dev| grep libffi-dev| grep ii
```

If the following information is displayed, the dependencies have been installed. Go to the next step. (The following information shows an example.)

```
gcc (Ubuntu 7.3.0-3ubuntu1~18.04) 7.3.0
GNU Make 4.7
```

Otherwise, run the following command to install the software. You can change the following command to install only some of them as required.

```
sudo apt-get install -y gcc g++ make cmake zlib1g zlib1g-dev openssl libsqlite3-dev libssl-dev libffi-dev unzip pcreutils net-tools libblas-dev gfortran libblas3 libopenblas-dev libncursesw5-dev
```

⚠️ CAUTION

- If you install dependencies as the root user, delete sudo from the commands in steps 1 and 2.
- If Python and its dependency are installed as a non-root user, run the su - username command to switch to the non-root user, and then perform step 1 to step 3.
- libsqlite3-dev must be installed before the Python installation. If the Python 3.7.5 environment has been installed in the user's OS and libsqlite3-dev is installed after the Python 3.7.5 environment is installed, you need to rebuild the Python environment.

```
sudo apt-get install -y gcc g++ make cmake zlib1g zlib1g-dev openssl libsqlite3-dev libssl-dev libffi-dev unzip pcreutils net-tools libblas-dev gfortran libblas3 libopenblas-dev libncursesw5-dev
```
Step 2 Check whether the Python development environment that meets the version requirements is installed in the system.

Run the python3.7 --version command. If the returned information meets the Python version requirements (3.7.0 to 3.7.9), go to the next step.

Otherwise, use the following procedure to install Python 3.7.5:

1. Run the following command to download the source code package of Python 3.7.5 to any directory of the installation environment.
   ```bash
   wget https://www.python.org/ftp/python/3.7.5/Python-3.7.5.tgz
   ```

2. Run the following command to go to the download directory and decompress the source code package:
   ```bash
   tar -zxvf Python-3.7.5.tgz
   ```

3. Go to the decompressed folder and run the following configuration, build, and installation commands:
   ```bash
   cd Python-3.7.5
   ./configure --prefix=/usr/local/python3.7.5 --enable-loadable-sqlite-extensions --enable-shared
   make
   sudo make install
   ```

   The --prefix parameter specifies the Python installation path. You can change it based on the site requirements. The --enable-shared parameter is used to compile the libpython3.7m.so.1.0 dynamic library. The --enable-loadable-sqlite-extensions parameter is used to load libsqlite3-dev.

   This document uses --prefix=/usr/local/python3.7.5 as an example. After the configuration, compilation, and installation commands are executed, the installation package is stored in the /usr/local/python3.7.5 directory, and the libpython3.7m.so.1.0 dynamic library is stored in the /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 directory.

4. Run the following commands to set the soft link:
   ```bash
   sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/local/python3.7.5/bin/python3.7.5
   sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/local/python3.7.5/bin/pip3.7.5
   ```

5. Set the Python 3.7.5 environment variables.
   ```bash
   # Set the Python 3.7.5 library path.
   export LD_LIBRARY_PATH=/usr/local/python3.7.5/lib:$LD_LIBRARY_PATH
   # If multiple Python 3 versions exist in the user environment, use Python 3.7.5.
   export PATH=/usr/local/python3.7.5/bin:$PATH
   ```

   To ensure that the python3.7.5 environment variables can be automatically configured when you install the CANN software package and run related services, create the use_private_python.info file in advance. The following is an example:
   
   Run the following command to create the file:

   1. User root:
      ```bash
      vi /etc/use_private_python.info
      ```

   2. Non-root user. The HwHiAiUser user is used as an example.
      ```bash
      vi /home/HwHiAiUser/use_private_python.info
      ```

   3. Add the following content to the file:
      ```bash
      python37_install_path=/usr/local/python3.7.5
      ```
      In the preceding command, /usr/local/python3.7.5 indicates the Python 3.7.5 installation path. Replace it as required.

   4. Run the :wq! command to save the file and exit.
6. After the installation is complete, run the following commands to check the installation version. If the required version information is displayed, the installation is successful.

   python3.7 --version
   pip3.7 --version

Step 3 Before the installation, run the pip3.7 list command to check whether the dependencies have been installed. If yes, skip this step. If not, run the following command to install the dependencies. If only some of the software is not installed, modify the following command to install selected software only.

   ● Configure the pip source before the installation. For details, see 10.5 Configuring the PIP Source.
   ● Run the pip3.7 --version command to query the version. If the version is earlier than 18.0, run the pip3.7 install --upgrade pip command to perform an upgrade.
   ● If you run the following commands as a non-root user, add --user at the end of each installation command, for example, pip3.7 install attrs --user. The installation command can be run in any path.
   ● The NumPy version must be 1.13.3 or later and earlier than 1.20. The recommended NumPy version is 1.17.2.

   pip3.7 install attrs
   pip3.7 install numpy==1.17.2
   pip3.7 install decorator
   pip3.7 install sympy
   pip3.7 install cffi
   pip3.7 install pyyaml
   pip3.7 install pathlib2
   pip3.7 install psutil
   pip3.7 install protobuf
   pip3.7 install scipy
   pip3.7 install requests

   If the error message "subprocess.CalledProcessError: Command '((lsb_release', '-a')' return non-zero exit status 1" is displayed, rectify the fault by referring to 12.7 Error Message "subprocess.CalledProcessError: Command '((lsb_release', '-a')' return non-zero exit status 1" Is Displayed During pip3.7 Installation.

-----End

7.3.2.4 Installation Procedure (CentOS 7.6)

Checking the Source Validity

The installation requires the download of related dependencies. Ensure that the installation environment can be connected to the network.

Run the following command as the root user to check whether the source is valid:

   yum makecache

If an error is reported during command execution or dependency installation, check whether the network is connected, or replace the source in the /etc/yum.repos.d/xxxx.repo file with an available source or use an image source. For details about how to configure a network proxy, see 10.3 Configuring a System Network Proxy.
If the message "Your license is invalid" is displayed after you run the preceding command, obtain the OS license.

Configuring Permissions for the Installation User

Skip the following part if you install as the root user.

Perform the following steps as the root user:

1. Run the following command to install sudo:
   ```
   yum install sudo
   ```

2. Open the `/etc/sudoers` file:
   ```
   chmod u+w /etc/sudoers
   vi /etc/sudoers
   ```

3. Add the following content under # User privilege specification in the file:
   ```
   username ALL=(ALL:ALL) NOPASSWD:SETENV:/usr/bin/yum, /usr/bin/pip, /bin/tar, /bin/mkdir, /bin/sh, /bin/bash, /usr/bin/make install, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/local/python3.7.5/bin/python3.7.5, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/local/python3.7.5/bin/pip3.7.5
   ```
   In the preceding command, `username` indicates the installation user. Change it as required.

   **CAUTION**
   - Ensure that the `/etc/sudoers` file contains `#includedir /etc/sudoers.d`. If the file does not contain `#includedir /etc/sudoers.d`, manually add it to the end of the file.
   - After the CANN software package has been installed, you can cancel the sudo permission.
   - When uninstalling or upgrading the CANN software package, you also need to configure the preceding user permissions.

   4. Run `:wq!` to save the file.

   5. Run the following command to revoke the write permission on the `/etc/sudoers` file:
   ```
   chmod u-w /etc/sudoers
   ```

Configuring the Maximum Number of Threads

1. Log in to the installation environment as the root user.
2. Configure environment variables to change the maximum number of threads.
   In the `/etc/profile` file, add the following content to the end of the file, save the file, and exit:
   ```
   ulimit -u unlimited
   ```
3. Run the following command to make the environment variable take effect:
   ```
   source /etc/profile
   ```

Installing Dependencies

**Step 1** Check whether the Python dependencies and GCC software are installed.
Run the following commands to check whether the dependencies such as GCC, Make, and Python are installed:

gcc --version
g++ --version
make --version
cmake --version
rpm -qa |grep unzip
rpm -qa |grep zlib-devel
rpm -qa |grep libffi-devel
rpm -qa |grep openssl-devel
rpm -qa |grep pcre
rpm -qa |grep net-tools
rpm -qa |grep sqlite-devel
rpm -qa |grep lapack-devel
rpm -qa |grep openblas-devel
rpm -qa |grep gcc-gfortran

If the following information is displayed, the dependencies have been installed. Go to the next step. (The following information shows an example.)

gcc (GCC) 4.8.5 20150623 (Red Hat 4.8.5-39)
g++ (GCC) 4.8.5 20150623 (Red Hat 4.8.5-39)
GNU Make 3.82
cmake version 2.8.12.2
unzip-6.0-21.el7.aarch64
zlib-devel-1.2.7-18.el7aarch64
libffi-devel-3.0.13-18.el7aarch64
openssl-devel-1.0.2k-19.el7aarch64
pcre-3.5.1-3.el7.aarch64
net-tools-2.0-0.25.20131004git.el7.aarch64
sqlite-devel-3.7.17-8.el7.aarch64
lapack-devel-3.4.2-8.el7.aarch64
openblas-devel-0.3.3-2.el7.aarch64
gcc-gfortran-4.8.5-39.el7.aarch64

Otherwise, run the following command to install the software. You can change the following command to install only some of them as required.

*** CAUTION ***

- If you install dependencies as the root user, delete sudo from the commands in step 1 to step 3.
- If Python and its dependency are installed as a non-root user, run the su - username command to switch to the non-root user, and then perform step 1 to step 3.
- sqlite-devel must be installed before the Python installation. If the Python 3.7.5 environment has been installed in the user's OS and sqlite-devel is installed after the Python 3.7.5 environment is installed, you need to rebuild the Python environment.

```bash
sudo yum install -y gcc gcc-c++ make cmake unzip zlib-devel libffi-devel openssl-devel pcre pcre-utils net-tools sqlite-devel lapack-devel openblas-devel gcc-gfortran
```

In the preceding steps, the message "No package libopenblas available" is displayed during the installation of openblas-devel. Run the following command to install the Linux extension package of the enterprise edition:

```bash
sudo yum install epel-release
```

If the CMake version installed using the preceding method is earlier than 3.5.1, see 10.4 Installing CMake 3.5.2.
Step 2  Check whether the Python development environment that meets the version requirements is installed in the system.

Run the `python3.7 --version` command. If the returned information meets the Python version requirements (3.7.0 to 3.7.9), go to the next step.

Otherwise, use the following procedure to install Python 3.7.5:

1. Run the following command to download the source code package of Python 3.7.5 to any directory of the installation environment.
   ```bash
   wget https://www.python.org/ftp/python/3.7.5/Python-3.7.5.tgz
   ```
2. Run the following command to go to the download directory and decompress the source code package:
   ```bash
   tar -zxvf Python-3.7.5.tar.gz
   ```
3. Go to the decompressed folder, create an installation directory, and run the configuration, build, and installation commands.
   ```bash
   cd Python-3.7.5
   ./configure --prefix=/usr/local/python3.7.5 --enable-loadable-sqlite-extensions --enable-shared
   make
   sudo make install
   ```
   --prefix specifies the Python installation path. You can modify it as required.
   --enable-shared is used to build the `libpython3.7m.so.1.0` dynamic library. --enable-loadable-sqlite-extensions is used to load the sqlite-devel dependency.

This document uses `--prefix=/usr/local/python3.7.5` as an example. After the configuration, compilation, and installation commands are executed, the installation package is stored in the `/usr/local/python3.7.5` directory, and the `libpython3.7m.so.1.0` dynamic library is stored in the `/usr/local/python3.7.5/lib/libpython3.7m.so.1.0` directory.

4. Run the following commands to set the soft link:
   ```bash
   sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/local/python3.7.5/bin/python3.7.5
   sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/local/python3.7.5/bin/pip3.7.5
   ```
5. Set the Python 3.7.5 environment variables.
   ```bash
   # Set the Python 3.7.5 library path.
   export LD_LIBRARY_PATH=/usr/local/python3.7.5/lib:$LD_LIBRARY_PATH
   # If multiple Python 3 versions exist in the user environment, use Python 3.7.5.
   export PATH=/usr/local/python3.7.5/bin:$PATH
   ```

### NOTE

To ensure that the python3.7.5 environment variables can be automatically configured when you install the CANN software package and run related services, create the `use_private_python.info` file in advance. The following is an example:

1. Run the following command to create the file:
   ```bash
   ▪ User root: vi /etc/use_private_python.info
   ▪ Non-root user. The HwHiAiUser user is used as an example. vi /home/HwHiAiUser/use_private_python.info
   ```

2. Add the following content to the file:
   ```bash
   python37_install_path=/usr/local/python3.7.5
   ```
   In the preceding command, `/usr/local/python3.7.5` indicates the Python 3.7.5 installation path. Replace it as required.

3. Run the `wq!` command to save the file and exit.
6. After the installation is complete, run the following commands to check the installation version. If the required version information is displayed, the installation is successful.

```
python3.7 --version
pip3.7 --version
```

**Step 3** Before the installation, run the `pip3.7 list` command to check whether the dependencies have been installed. If yes, skip this step. If not, run the following command to install the dependencies. If only some of the software is not installed, modify the following command to install selected software only.

- Configure the pip source before the installation. For details, see **10.5 Configuring the PIP Source**.
- Run the `pip3.7 --version` command to query the version. If the version is earlier than 18.0, run the `pip3.7 install --upgrade pip` command to perform an upgrade.
- If you run the following commands as a non-root user, add `--user` at the end of each installation command, for example, `pip3.7 install attr --user`. The installation command can be run in any path.
- The NumPy version must be 1.13.3 or later and earlier than 1.20. The recommended NumPy version is 1.17.2.

```
pip3.7 install attr
pip3.7 install numpy==1.17.2
pip3.7 install decorator
pip3.7 install sympy
pip3.7 install pyyaml
pip3.7 install pathlib2
pip3.7 install requests
```

- If an error is reported during the NumPy installation, rectify the fault by referring to **12.6 pip3.7 install numpy Error**.
- If an error is reported during the SciPy installation, rectify the fault by referring to **12.5 pip3.7 install scipy Error**.

**Step 4** Skip this step if only offline inference is required.

CentOS 7.6 uses GCC 4.8.5 by default. For details about how to install GCC 7.3.0, see **10.6 Installing GCC 7.3.0**.

----End

### 7.3.3 Installing the Deep Learning Engine Package

**Prerequisites**

- Before installing the software, ensure that the Ascend AI Processor driver has been installed in the installation environment.
- The deep learning engine package `Ascend-cann-nnae_{version}_linux-{arch}.run` has been obtained. For details, see **7.2 Obtaining Software Packages**.
**Procedure**

To obtain and install the deep learning engine package, perform the following steps:

**Step 1** Log in to the installation environment as the installation user of the software package.

If the dependency installation user in 7.3.2 Installing OS Dependencies is the root user, you can specify an installation user of the software package. If the dependency installation user in 7.3.2 Installing OS Dependencies is a non-root user, ensure that the installation user of the software package is the same as the dependency installation user.

**Step 2** Upload the deep learning engine package to any directory (for example, /home) in the installation environment.

**Step 3** Go to the directory where the software package is stored.

**Step 4** Grant the execute permission on the software package.

```sh
cat +x *.run
```

*NOTE*  
*.run indicates the software package name, for example, the deep learning engine package `Ascend-cann-nnae_{version}_linux-{arch}.run`. Replace it with the actual package name.

**Step 5** Run the following command to check the consistency and integrity of the software package installation file:

```sh
./*.run --check
```

**Step 6** Install the software. (The following commands support --install-for-all and --install-path=<path>. For details about the parameters, see 13.1 Parameters.)

- If the installation is performed as a non-root user, run the following command:

  ```sh
  ./*.run --install
  ```

- If the installation is performed as the root user:
  - As the default running user HwHiAiUser, run the following command:

    ```sh
    ./*.run --install
    ```
  - As a specified running user, run the following command:

    ```sh
    ./*.run --install-username=username --install-usergroup=usergroup --install
    ```

  where, `--install-username` and `--install-usergroup` are used to specify the running users.
7.3.4 Post-installation Operations

7.3.4.1 Installing the Framework Plugin Package

If the deep learning framework TensorFlow is used for online inference or training, you need to install the framework plugin package `Ascend-cann-tfplugin_xxx.run`.

Install the framework plugin package by referring to 7.3.3 Installing the Deep Learning Engine Package.

7.3.4.2 Installing the Deep Learning Framework

A deep learning framework needs to be installed in online inference or training scenarios.

This document uses TensorFlow as an example to describe how to install the deep learning framework. If you want to use the MindSpore framework, visit [https://www.mindspore.cn/install/en](https://www.mindspore.cn/install/en) to obtain the method of installing the MindSpore framework. If you want to use the PyTorch framework, see "Installing the PyTorch Framework" in the *PyTorch Network Model Porting and Training Guide*.

Installation Preparations

- For the x86 architecture, skip the installation preparations.
- For the AArch64 architecture:
  - TensorFlow depends on h5py, and h5py depends on HDF5. Therefore, you need to compile and install HDF5 first. Otherwise, an error is reported when you use pip to install h5py. Perform the following operations as the root user.
- TensorFlow depends on grpcio. Therefore, you are advised to install the grpcio of the specified version before installing TensorFlow. Otherwise, the installation may fail.

**Step 1** Compile and install the HDF5.

1. Run the following command to download the source code package of HDF5 to any directory of the installation environment:
   ```bash
   wget https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.5/src/hdf5-1.10.5.tar.gz --no-check-certificate
   ```
2. Run the following command to go to the download directory and decompress the source code package:
   ```bash
   tar -zxvf hdf5-1.10.5.tar.gz
   ```
   Go to the decompressed folder and run the following configuration, build, and installation commands:
   ```bash
   cd hdf5-1.10.5/
   ./configure --prefix=/usr/include/hdf5
   make
   make install
   ```

**Step 2** Configure environment variables and create a soft link to the DLL.

1. Configure environment variables.
   ```bash
   export CPATH="/usr/include/hdf5/include/:/usr/include/hdf5/lib/"
   ```
2. Run the following commands as the root user to create a soft link to the DLL. Add sudo before the following commands as a non-root user:
   ```bash
   ln -s /usr/include/hdf5/lib/libhdf5.so  /usr/lib/libhdf5.so
   ln -s /usr/include/hdf5/lib/libhdf5_hl.so  /usr/lib/libhdf5_hl.so
   ```

**Step 3** Install h5py.

Run the following command as the root user to install the h5py dependency:
```bash
pip3.7 install Cython
```
The h5py installation command is as follows:
```bash
pip3.7 install h5py==2.8.0
```

**Step 4** Install grpcio.

Run the following command as the root user to install grpcio:
```bash
pip3.7 install grpcio==1.32.0
```

---End

**Installing TensorFlow 1.15.0**

TensorFlow 1.15 is necessary to develop and verify operators and develop training services.

- For the x86 architecture, download the software package from the pip source. For details, see [https://www.tensorflow.org/install/pip?lang=python3](https://www.tensorflow.org/install/pip?lang=python3).
  Note that the instructions provided by the TensorFlow website are incorrect. To download the CPU version from the pip source, you need to explicitly specify `tensorflow-cpu`. Otherwise, the GPU version is downloaded by default. That is, change `tensorflow==1.15 — Release for CPU-only` to `tensorflow-cpu==1.15 — Release for CPU-only`. In addition, the installation command `pip3 install --user --upgrade tensorflow` described on the official website needs to be changed to `pip3.7 install Tensorflow-cpu==1.15` as the
For the AArch64 architecture, perform the following operations:

a. Download the compiled TensorFlow 1.15 software package:
   `tensorflow-1.15.0-cp37-cp37m-linux_aarch64.whl`

b. Install the compiled TensorFlow.

   Go to the directory where the software package is stored and run the following command as the root user to install TensorFlow 1.15:
   ```
   pip3.7 install tensorflow-1.15.0-cp37-cp37m-linux_aarch64.whl
   ```

   Run the following command as a non-root user:
   ```
   pip3.7 install tensorflow-1.15.0-cp37-cp37m-linux_aarch64.whl --user
   ```

### 7.3.4.3 Installing protobuf Python

If the training script depends on the Python version of protobuf to store data in the serialized structure (for example, the serialization interfaces of TensorFlow), you need to install protobuf Python.

**Step 1** Check whether the system contains the dynamic library `/usr/local/python3.7.5/lib/python3.7/site-packages/google/protobuf/pyext/_message.cpython-37m-<arch>-linux-gnu.so` (Python 3.7.5 installed by referring to 7.3.2 Installing OS Dependencies is used as an example). If not, perform the following steps to install it: `<arch>` indicates the system architecture type.

**NOTE**

```
/usr/local/python3.7.5/lib/python3.7/site-packages` is the path for installing third-party libraries using pip. You can run the `pip3.7 -V` command to query the path.
```

If the system displays `/usr/local/python3.7.5/lib/python3.7/site-packages/pip`, the path for installing third-party libraries using pip is `/usr/local/python3.7.5/lib/python3.7/site-packages`.

**Step 2** Uninstall protobuf.

```
pip3.7 uninstall protobuf
```

**Step 3** Download the protobuf software package.

Download the `protobuf-python-3.11.3.tar.gz` software package (or another version that is compatible with TensorFlow installed in the current environment) from [https://github.com/protocolbuffers/protobuf/releases/download/v3.11.3/protobuf-python-3.11.3.tar.gz](https://github.com/protocolbuffers/protobuf/releases/download/v3.11.3/protobuf-python-3.11.3.tar.gz), upload the package to any directory on the Linux server as the root user, and run the `tar zxvf protobuf-python-3.11.3.tar.gz` command to decompress the package.

**Step 4** Install protobuf as the root user.

Go to the protobuf software package directory.

1. Install the dependencies of protobuf.

   If the OS is Ubuntu, run the following command:
   ```
   apt-get install autoconf automake libtool curl make g++ unzip libffi-dev
   ```

   If the OS is EulerOS, run the following command:
   ```
   yum install autoconf automake libtool curl make gcc-c++ unzip libffi-devel
   ```

   If the OS is CentOS or BCLinux, run the following command:
yum install autoconf automake libtool curl make gcc-c++ unzip libffi-devel -y

2. Grant the execute permission on the `autogen.sh` script and execute the script.
   ```
   chmod +x autogen.sh
   ./autogen.sh
   ```

3. Configure the installation path. The default installation path is `/usr/local`.
   ```
   ./configure
   ```
   To specify the installation path, run the following command:
   ```
   ./configure --prefix=/protobuf
   ```
   `/protobuf` indicates the installation path specified by the user.

4. Run the following commands to install protobuf:
   ```
   make -j15 # Check the number of CPUs by running grep -w processor /proc/cpuinfo|wc -l
   ```
   In this example, the number is 15. You can set the parameters as required.
   ```
   make install
   ```

5. Refresh the shared library.
   ```
   ldconfig
   ```
   After protobuf is installed, a `google/protobuf` folder is generated in the `include` directory under the path configured in Step 4.3 to store the related header files of protobuf. In addition, a `.protoc` executable file is generated in the `bin` directory under the path configured in Step 4.3. The `.protoc` executable file is used to build the `.proto` file and generate the C++ header file and implementation file of protobuf.

6. Check whether the software package is successfully installed.
   ```
   ln -s /protobuf/bin/protoc /usr/bin/protoc
   ```
   ```
   protoc --version
   ```
   `/protobuf` is the installation path configured in Step 4.3. If the installation path is not configured, run the `protoc --version` command to check whether the installation is successful.

**Step 5** Install the runtime library of protobuf Python.

1. Go to the `python` subdirectory in the protobuf software package directory and compile the Python runtime library.
   ```
   python3.7 setup.py build --cpp_implementation
   ```
   **NOTE**
   Build a runtime library of the binary version. If the runtime library of the binary version cannot be generated by running the `python3.7 setup.py build` command, the processing of the serialization structure is slow.

2. Install the dynamic library.
   ```
   cd .. && make install
   ```
   Go to the `python` subdirectory and install the Python runtime library.
   ```
   python3.7 setup.py install --cpp_implementation
   ```

3. Check whether the library has been installed.
   Check whether the dynamic library `/usr/local/python3.7.5/lib/python3.7/site-packages/protobuf-3.11.3-py3.7-linux-aarch64.egg/google/protobuf/`
pyext/_message.cpython-37m-<arch>-linux-gnu.so exists in the system.
<arch> indicates the system architecture type.

**NOTE**

/usr/local/python3.7.5/lib/python3.7/site-packages is the path for installing third-party libraries using pip. You can run the `pip3.7 -V` command to query the path.

If the system displays `/usr/local/python3.7.5/lib/python3.7/site-packages/pip`, the path for installing third-party libraries using pip is `/usr/local/python3.7.5/lib/python3.7/site-packages`.

4. If you have specified the installation path in **Step 4.3**, add the following environment variable settings to the run script:

```bash
export LD_LIBRARY_PATH=/protobuf/lib:${LD_LIBRARY_PATH}
```

In the preceding command, `/protobuf` indicates the installation path set by the user in **Step 4.3**.

5. Establish a soft link.

If you configure the installation path, you need to establish a soft link. Otherwise, an error will be reported when TensorFlow is imported. The command is as follows:

```bash
ln -s /protobuf/lib/libprotobuf.so.22.0.3 /usr/lib/libprotobuf.so.22
```

`/protobuf` is the installation path configured in **Step 4.3**.

---End

### 7.3.4.4 Configuring Environment Variables

The CANN software provides a script for setting process-level environment variables. You can reference the script in processes to automatically configure environment variables. The environment variables automatically become invalid after the processes end. The default installation path of the root user is used as an example.

```bash
# Configure environment variables when installing the nnae package.
./usr/local/Ascend/nnae/set_env.sh
```

```bash
# Configure environment variables when installing the tfplugin package.
./usr/local/Ascend/tfplugin/set_env.sh
```

You can also configure permanent environment variables by modifying the `~/.bashrc` file. The procedure is as follows:

1. Run the `vi ~/.bashrc` command in any directory as the running user to open the `~/.bashrc` file and append the preceding lines to the file.
2. Run the `:wq!` command to save the file and exit.
3. Run the `source ~/.bashrc` command for the modification to take effect immediately.

### 7.3.4.5 Configuring the NIC IP Address of a Device

During distributed training, you need to use the HCCN Tool in the Ascend software to configure the NIC IP address of the device. This IP address is used for communication between devices to synchronize network model parameters. This section describes only the commands for configuring the network by using the HCCN tool. If you need to use other functions of the HCCN tool (for example,
checking the link status of a network port), see the *Ascend 910 HCCN Tool Interface Reference*.

**Atlas 800 training server and Atlas 900 AI cluster**

**NOTE**

To check whether the SMP or AMP mode is used, log in to the BMC background and run the `ipmcget -d npuworkmode` command.

- **SMP (symmetric multi-processor) mode**
  
  Log in to the AI Servers as the `root` user and configure the NIC IP address of each device. The configuration requirements are as follows:
  
  - NICs 0 and 4, 1 and 5, 2 and 6, and 3 and 7 of an AI Server must be in the same network segment respectively. NICs 0, 1, 2, and 3 must be in different network segments. NICs 4, 5, 6, and 7 must be in different network segments.
  
  - In the cluster scenario, the devices in the similar positions on AI Servers must be in the same network segment. For example, NIC 0 of AI Server 1 and AI Server 2 must be in the same network segment, and NIC 1 of AI Server 1 and AI Server 2 must be in the same network segment. Change the IP address as required.

  ```
  hccn_tool -i 0 -ip -s address 192.168.100.1 netmask 255.255.255.0
  hccn_tool -i 1 -ip -s address 192.168.101.1 netmask 255.255.255.0
  hccn_tool -i 2 -ip -s address 192.168.102.1 netmask 255.255.255.0
  hccn_tool -i 3 -ip -s address 192.168.103.1 netmask 255.255.255.0
  hccn_tool -i 4 -ip -s address 192.168.100.2 netmask 255.255.255.0
  hccn_tool -i 5 -ip -s address 192.168.101.2 netmask 255.255.255.0
  hccn_tool -i 6 -ip -s address 192.168.102.2 netmask 255.255.255.0
  hccn_tool -i 7 -ip -s address 192.168.103.2 netmask 255.255.255.0
  ```

- **AMP (asymmetric multi-processor) mode**
  
  In AMP mode, you do not need to configure the NIC IP address of the device.

**Atlas 300T training card**

Each server can be configured with one Atlas 300T training card or two. Each card corresponds to one Device OS and needs to be configured with one IP address. Different cards need to be configured with IP addresses in the same network segment.

Log in to the AI Servers as the `root` user and configure the NIC IP address of each device. The configuration operations are as follows:

**Step 1** Run the `npu-smi info` command to view the ID of the device to be configured. In Figure 7-2, the NPU IDs are 1 and 4, for example. Use the actual NPU IDs in the query result.

**Figure 7-2 Checking the device ID**

![Figure 7-2 Checking the device ID](image)
Step 2 Run the following commands to configure the NIC IP addresses of the device. The IP addresses used in the following example are for reference only.

```
hccn_tool -i 1 -ip -s address 192.168.0.2 netmask 255.255.255.0
hccn_tool -i 4 -ip -s address 192.168.0.3 netmask 255.255.255.0
```

---End

**NOTE**

- Ensure that the npu-smi tool has been installed on the server.
- In cluster scenarios, the devices of each AI server must be in the same network segment.

### 7.3.4.6 Follow-up

- Port models. For details, see the *CANN TensorFlow Network Model Porting and Training Guide* or *PyTorch Network Model Porting and Training Guide*.
- Obtain an offline model or training script from ModelZoo.
8.1 Before You Start

8.2 Upgrade Operations

8.1 Before You Start

Upgrade Impacts
- Do not perform other maintenance operations during the upgrade.
- Services are interrupted during the software upgrade.
- After the software package is upgraded, services are not affected.

Precautions

Table 8-1 describes the precautions for the upgrade.

<table>
<thead>
<tr>
<th>No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Before the upgrade, read this document carefully to ensure that you have learned all the content. For any problems or suggestions pertaining to the document, contact Huawei technical support.</td>
</tr>
<tr>
<td>2</td>
<td>To reduce the impact on services, switch services to other nodes or perform the upgrade during off-peak hours.</td>
</tr>
<tr>
<td>3</td>
<td>After the upgrade, ensure that the versions of the components are consistent.</td>
</tr>
<tr>
<td>4</td>
<td>If there are multiple devices, you can also perform the upgrade. By default, only the version pointed to by the latest soft link in the installation directory (that is, the current version) is upgraded.</td>
</tr>
</tbody>
</table>
Version Requirements

The driver version and firmware version must match the CANN software version. For details about the version mapping, see the version mapping table in the CANN.

Table 8-2 Upgrade description

<table>
<thead>
<tr>
<th>Source Version</th>
<th>Target Version</th>
<th>Upgrade Supported or Not</th>
</tr>
</thead>
<tbody>
<tr>
<td>CANN 3.0.0</td>
<td>CANN 5.0.2</td>
<td>No. Perform the uninstallation and then installation.</td>
</tr>
<tr>
<td>CANN 3.1.0</td>
<td>CANN 5.0.2</td>
<td></td>
</tr>
<tr>
<td>CANN 3.2.0</td>
<td>CANN 5.0.2</td>
<td></td>
</tr>
<tr>
<td>CANN 3.2.1</td>
<td>CANN 5.0.2</td>
<td>Yes.</td>
</tr>
<tr>
<td>CANN 3.3.0</td>
<td>CANN 5.0.2</td>
<td>Yes.</td>
</tr>
</tbody>
</table>

Upgrade Process

Upgrade the firmware, driver, and CANN software in sequence.

8.2 Upgrade Operations

Upgrading the Driver and Firmware

Perform the upgrade operations by referring to Table 8-3.

Table 8-3 Driver and firmware upgrade guide

<table>
<thead>
<tr>
<th>Product Model</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atlas 200 AI accelerator module (model 3000)</td>
<td>Atlas 200 AI Accelerator Module 1.0.7 or Later Software Installation and Maintenance Guide (EP, Model 3000)</td>
</tr>
<tr>
<td>Atlas 300i inference card (model 3000)</td>
<td>Atlas 300i Inference Card 1.0.10 Upgrade Guide (Models 3000, 3010)</td>
</tr>
<tr>
<td>Atlas 300i inference card (model 3010)</td>
<td>Atlas 300i Inference Card 1.0.10 Upgrade Guide (Models 3000, 3010)</td>
</tr>
<tr>
<td>Atlas 500 Pro AI edge server (model 3000)</td>
<td>An Atlas 300i inference card (model 3000) is supported. For details, see the Atlas 300i Inference Card 1.0.10 Upgrade Guide (Models 3000, 3010).</td>
</tr>
</tbody>
</table>
## Upgrading the CANN Software

The upgrade operations of CANN software are as follows:

**Step 1** Log in to the installation environment as the installation user of the software package.

**Step 2** Go to the directory where the software package is stored.

**Step 3** Grant the execute permission on the software package.

```
chmod +x software package name.run
```

**Step 4** Run the following command to check the consistency and integrity of the software package installation file:

```
./software package name.run --check
```

**Step 5** Upgrade the software.

- Upgrade CANN 3.2.1 to the current version.
  - If the installation path is specified during the installation, run the following command:
    
    ```
    ./software package name.run --upgrade --install-path=<path>
    ```

    In the preceding command, `<path>` indicates the specified installation directory of the software package. Replace the software package name with the actual one.

- If the installation path is not specified during the installation, run the following command:

```
When CANN 3.3.0 is upgraded to a later version, the original configuration information can be automatically imported from the installation configuration file, and no installation parameter needs to be entered. Run the following command:

```
$software package name run --upgrade
```

**NOTE**

If a message is displayed during the upgrade indicating that a non-root user does not have related permissions (such as the rm or cp permission), configure related permissions for the user by referring to *Configuring Permissions for the Installation User*. After the upgrade is complete, revoke the permissions.

----End
Uninstallation

- If you uninstall only the CANN software package (such as nnrt and ascend-toolkit), you can uninstall them in any sequence. However, if you also need to uninstall the driver and firmware, you need to uninstall other software packages and then uninstall the driver and firmware.

- If a message is displayed during the uninstallation indicating that a non-root user does not have related permissions (such as the rm or cp permission), configure related permissions for the user by referring to Configuring Permissions for the Installation User. After the uninstallation is complete, revoke the permissions.

Method 1: Using a Script

You can use the uninstallation script to uninstall a software package. The following describes how to uninstall the development kit package.

Step 1 Go to the directory where the uninstallation script of the software is stored. Generally, the script is stored in the script directory.

```
cd <path>/ascend-toolkit/<version>/xxx-linux/script
```

<path> indicates the tool installation directory. If the software package is installed in a specified directory, replace it with the specified directory. If this parameter is not specified, the default directory is used.

<version> indicates the software package version and xxx-linux indicates the CPU architecture. Replace them as required.

Step 2 Run the ./uninstall.sh command to run the script.

-----End

Method 2: Uninstalling a Software Package

To uninstall an installed software package, perform the following steps:

Step 1 Log in to the installation environment as the installation user of the software package.

Step 2 Go to the directory where the software package is stored.
Step 3  Run the following commands to uninstall the software package:

```
./software package name run --uninstall
```

If the following information is displayed, the software is successfully uninstalled:

```
[INFO] xxx uninstall success
```

`xxx` indicates the name of the software package to be uninstalled.

----End
10 Common Operations

10.1 Installing the CANN Software Package (.deb)

The CANN software package in .deb format applies only to the Debian OS.

If the *.deb package is obtained, the installation procedure is as follows:

1. Configure permissions for the installation user. If you install the software package as a non-root user, perform this step. If you install the software package as the root user, skip this step.
   Configure the permission of the installation user as the root user and add /usr/bin/dpkg to the username of the installation user based on Configuring Permissions for the Installation User.

2. Log in to the server as the installation user.

<table>
<thead>
<tr>
<th>CAUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run the following command to switch from the root user to a non-root user: su - username</td>
</tr>
</tbody>
</table>

3. (Optional) Configure a running user.
   - To allow any user to run the script, run the following command as the installation user:
export ASCEND_INSTALL_FOR_ALL=true
- To specify a running user, run the following command as the installation user (if no running user is specified, the default running user is HwHiAiUser):
  export ASCEND_USER_NAME=username
  export ASCEND_USER_GROUP=usergroup

NOTE
Ensure that the owner group of the running user of the CANN software package is the same as that of the user who runs the driver. If the owner groups are different, add the user to the group of the driver running user. Example command:
  usermod -a -G user group username

4. Install the .deb package. (Replace the package name with the actual one.)
- If you install the package as the root user, run the following command:
  dpkg -i *.deb
- If you install the package as a non-root user, run the following command:
  sudo -E dpkg -i *.deb

CAUTION
If the .deb package fails to be installed and cannot be installed again, rectify the fault by referring to 12.9 How Do I Handle a .deb Package Installation Failure?.

10.2 Configuring the NIC IP Address

After the OS is installed, you need to configure the IP address of the NIC on the iBMC remote management page to remotely connect to the server. CentOS and Ubuntu are used as examples.

Configuring the NIC IP Address on CentOS

Step 1 Log in to the OS as the root user.

Step 2 Go to the configuration file of the NIC for which an IP address needs to be configured. NIC enp2s0f0 is used as an example. The NIC and path vary depending on the site requirements.

  vi /etc/sysconfig/network-scripts/ifcfg-enp2s0f0

Step 3 Configure the service IP address, subnet mask, and gateway of the corresponding NIC, as shown in Figure 10-1.
Figure 10-1 Setting network parameters

```
TYPE=Ethernet
PROTO=none
BOOTPROTO=none
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=eno16777730
UUID=71b3c94b-e746-44f8-a2e8-21b452746db0
DEVICE=eno16777730
ONBOOT=yes
IPADDR=10.174.28.173
PREFIX=22
GATEWAY=10.174.28.1
DNS1=10.129.2.34
IPV6_PRIVACY=none
```

**NOTE**

- **BOOTPROTO** indicates the IP address type of the device. For a static IP address, set this parameter to **static** or **none**. For a dynamic IP address, set this parameter to **dhcp** so that the IP address can be obtained automatically.
- Set **ONBOOT** to **yes** to enable automatic network connection.
- **PREFIX** indicates the network bit. If the value is **22**, the subnet mask is 255.255.252.0.

**Step 4** Restart the network service.

```bash
service network restart
```

**Step 5** Check whether the IP address is configured successful.

```bash
ifconfig
```

**Step 6** Run the following command to check the service status:

```bash
systemctl status NetworkManager
```

If the command output is **disabled**, run the following command to enable NetworkManager automatic startup:

```bash
systemctl enable NetworkManager
```

Check the service status again. If the command output is **enabled**, the configuration is successful.

--- End
Configuring the NIC IP Address on Ubuntu

**Step 1** Log in to the OS as the root user.

**Step 2** Go to the configuration file of the NIC. NIC enp125s0f0 is used as an example. The NIC and path vary depending on the site requirements.

```
vi /etc/netplan/01-netcfg.yaml
```

**Step 3** Configure the service IP address, subnet mask, and gateway, as shown in Figure 10-2.

![Figure 10-2 Configuring the IP address](image)

**Step 4** Apply the configuration file.

```
netplan apply
```

**Step 5** Check whether the IP address is configured successful.

```
ifconfig
```

----End

10.3 Configuring a System Network Proxy

The following procedure is a general method for configuring a network proxy. It may not be applicable to all network environments. The method of configuring the network proxy depends on the actual network environment.

**Prerequisites**

- Ensure that the network cable of the server is connected and the proxy server can connect to the external network.
- The configuration proxy is based on the condition that the server is located on an intranet and cannot be directly connected to the external network.

**Configuring a System Network Proxy**

**Step 1** Log in to the user environment as the root user.

**Step 2** Run the following command to edit the `/etc/profile` file:

```
vim /etc/profile
```
Add the following content to the file, save the file, and exit:

```
export http_proxy="http://user:password@proxyserverip:port"
export https_proxy="http://user:password@proxyserverip:port"
```

In the preceding commands, **user** indicates the username on the intranet, **password** (special characters need to be converted) indicates the user password, **proxyserverip** indicates the IP address of the proxy server, and **port** indicates the port number.

**Step 3** Run the following command to make the configuration take effect.

```
source /etc/profile
```

**Step 4** Run the following command to check whether the external network is connected:

```
wget www.baidu.com
```

If the HTML file can be downloaded, the server is connected to the external network successfully.

**NOTE**

If a certificate error occurs when you use a proxy to connect to the network, install the certificate of the proxy server before downloading third-party components.

----End

### 10.4 Installing CMake 3.5.2

1. Run the **wget** command to download the source code package of CMake to any directory on the server:
   
   ```
wget https://cmake.org/files/v3.5/cmake-3.5.2.tar.gz --no-check-certificate
   ```

2. Run the following command to go to the download directory and decompress the source code package:
   
   ```
tar -zxvf cmake-3.5.2.tar.gz
   ```

3. Go to the decompressed folder and run the following configuration, compilation, and installation commands:
   
   ```
   cd cmake-3.5.2
   ./bootstrap --prefix=/usr
   make
   sudo make install
   ```

4. After the installation is complete, run the **cmake --version** command again to check the version number.

### 10.5 Configuring the PIP Source

Configure the pip source as follows:

**Step 1** Run the following command as the installation user of the software package:

```
cd ~/.pip
```

If a message indicating that the directory does not exist is displayed, run the following commands to create and go to the directory:

```
mkdir ~/.pip
cd ~/.pip
```

**Step 2** Edit the **pip.conf** file.
Run the `vi pip.conf` command to open the `pip.conf` file and edit the file as follows.

```
[global]
# Huawei source is used as an example. Replace it based on the actual situation.
index-url = https://mirrors.huaweicloud.com/repository/pypi/simple
trusted-host = mirrors.huaweicloud.com
timeout = 120
```

**Step 3** Run the `:wq!` command to save the file and exit.

----End

## 10.6 Installing GCC 7.3.0

Perform the following steps as the `root` user:

**Step 1** Download `gcc-7.3.0.tar.gz` from [https://mirrors.tuna.tsinghua.edu.cn/gnu/gcc/gcc-7.3.0/gcc-7.3.0.tar.gz](https://mirrors.tuna.tsinghua.edu.cn/gnu/gcc/gcc-7.3.0/gcc-7.3.0.tar.gz).

**Step 2** The GCC installation occupies a large amount of temporary space. Therefore, run the following command to clear the `/tmp` directory:

```
sudo rm -rf /tmp/*
```

**Step 3** Install dependencies.

For CentOS, run the following command:

```
yum install bzip2
```

For Ubuntu, run the following command:

```
apt-get install bzip2
```

**Step 4** Compile and install GCC.

1. Go to the directory where the `gcc-7.3.0.tar.gz` source code package is located and run the following command to decompress the source code package:

```
tar -zxvf gcc-7.3.0.tar.gz
```

2. Go to the decompressed folder and download the GCC dependencies:

```
cd gcc-7.3.0
./contrib/download_prerequisites
```

If an error is reported during the execution of the preceding command, run the following commands to download the dependencies from the `gcc-7.3.0/` folder:

```
wget http://gcc.gnu.org/pub/gcc/infrasstructure/gmp-6.1.0.tar.bz2
wget http://gcc.gnu.org/pub/gcc/infrasstructure/mpc-1.0.3.tar.gz
wget http://gcc.gnu.org/pub/gcc/infrasstructure/isl-0.16.1.tar.bz2
```

After the preceding dependencies are downloaded, run the following command again:

```
./contrib/download_prerequisites
```

If the preceding command fails, ensure that the dependencies are successfully downloaded at a time and no repeated download occurs.

3. Run the configuration, compilation, and installation commands.

```
./configure --enable-languages=c,c++ --disable-multilib --with-system-zlib --prefix=/usr/local/gcc7.3.0
make -j15 # Check the number of CPUs by running `grep -w processor /proc/cpuinfo | wc -l`. In this example, the number is 15. You can set the parameters as required.
makesh
```

```
The **--prefix** parameter is used to specify the GCC 7.3.0 installation path. You can set it as required. However, do not set it to */usr/local* or */usr* because they conflict with the GCC that is installed by default by the system using the software source. Otherwise, the original GCC compilation environment of the system will be damaged. In this example, use */usr/local/gcc7.3.0*.

--- End

10.7 Querying the CANN Software Package Version

**Step 1** Log in to the installation environment as the installation user of the software package.

**Step 2** Go to the directory where the software package installation information file is stored. (The following uses the Ascend-cann-toolkit software package as an example.)

```
cd /usr/local/Ascend/ascend-toolkit/latest/{arch}-linux
```

In the preceding command, */usr/local/Ascend* is the default installation path of the *root* user. Replace it with the actual installation path. *{arch}* indicates the CPU architecture (ARM64 or x86_64).

**Step 3** Run the following command to query the version:

```
cat ascend_toolkit_install.info
```

--- End

10.8 Setting User Account Validity Period

To ensure user security, you need to set the validity period of a user. You can run the `chage` command to set the validity period of a user.

Command:

```
```

**Table 10-1** describes the parameters.

**Table 10-1** Setting user account validity period

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-m</td>
<td>Minimum time (in days) for which the password must be used. The password cannot be changed during this period. The value 0 indicates that the password can be changed at any time.</td>
</tr>
<tr>
<td>-M</td>
<td>Maximum validity period (days) of a password. The value -1 indicates that the validity check of the password can be disabled. The value 99999 indicates that the validity period is unlimited.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>-d</td>
<td>Date when the password was changed the last time.</td>
</tr>
<tr>
<td>-I</td>
<td>Maximum idle period (in days) after which the user account will be disabled. After the specified time period has expired, the password will be invalid.</td>
</tr>
<tr>
<td>-E</td>
<td>Date when the user account expires. The user account is unavailable when the account validity period has expired.</td>
</tr>
<tr>
<td>-W</td>
<td>Number of days in advance users are notified that their passwords are about to expire.</td>
</tr>
<tr>
<td>-l</td>
<td>Lists the current settings. It helps non-privileged users to determine the time when their passwords or accounts expire.</td>
</tr>
</tbody>
</table>

**NOTE**

- *Table 10-1* lists only common parameters. You can run the `chage --help` command to display detailed parameter description.
- The date format is YYYY-MM-DD. For example, `chage -E 2017-12-01 test` indicates that the password of user test expires on December 1, 2017.
- User is mandatory. Replace it with the actual user name. The default user name is root.

For example, to change the validity period of user *test* to December 31, 2017, run the following command:

```
chage -M 90 test
```
# 11 Software Packages

## Table 11-1 Overview of the software packages

<table>
<thead>
<tr>
<th>Package</th>
<th>Subpackage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development toolchain platform</td>
<td>MindStudio_*.tar.gz</td>
<td>It provides a graphical development interface for application development, debugging, model conversion, and network porting, optimization, and profiling.</td>
</tr>
<tr>
<td>Driver and firmware package</td>
<td>Axxx*-npu_*.zip</td>
<td>Driver and firmware installation package.</td>
</tr>
<tr>
<td>Development kit</td>
<td>Ascend-cann-toolkit_*</td>
<td>It is used to build and run applications, enabling AscendCL layer and GE model loading and execution functions. It contains the AscendCL libraries and building dependencies (excluding the libraries in the driver package).</td>
</tr>
<tr>
<td></td>
<td>Ascend-acllib</td>
<td>pyACL installation package, including the Python library and sample code required for pyACL-based development.</td>
</tr>
<tr>
<td></td>
<td>Ascend-atc</td>
<td>It is mainly used for offline model conversion.</td>
</tr>
<tr>
<td>Package</td>
<td>Subpackage</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Ascend-opp</td>
<td></td>
<td>Operator library, including operator prototype libraries, operator implementation libraries, operator plugins, and fusion patterns. The operator implementation library includes TBE operators and AI CPU operators.</td>
</tr>
<tr>
<td>Ascend-toolkit</td>
<td></td>
<td>Commissioning tool kit, mainly including the tools that the developer needs to use for the test application and the operator.</td>
</tr>
<tr>
<td>Ascend-fwkacllib</td>
<td></td>
<td>FwkACLlib supports offline inference, online inference, and training. Fwklib deliverables used to build graphs and operators online and load computational graphs to the NPU for execution, supporting training, offline model conversion, custom operator development, and fusion pattern development.</td>
</tr>
<tr>
<td>aicpu_kernels</td>
<td></td>
<td>Operator package, providing the kernel implementation of AI CPU operators. For details about AI CPU kernel loading, see 13.3 AI CPU Kernel Loading Description.</td>
</tr>
<tr>
<td>Ascend-test_ops</td>
<td></td>
<td>Test operator package, which is used to test the computing power and power consumption of Ascend AI Processors.</td>
</tr>
<tr>
<td>Package</td>
<td>Subpackage</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Offline inference engine package</td>
<td>Ascend-acllib</td>
<td>It is used to build and run applications, enabling AscendCL layer and GE model loading and execution functions. It contains the AscendCL libraries and building dependencies (excluding the libraries in the driver package).</td>
</tr>
<tr>
<td>Ascend-cann-nnrt_*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ascend-pyACL</td>
<td></td>
<td>pyACL installation package, including the Python library and sample code required for pyACL-based development.</td>
</tr>
<tr>
<td>aicpu_kernels</td>
<td></td>
<td>Operator package, providing the kernel implementation of AI CPU operators. For details about AI CPU kernel loading, see 13.3 AI CPU Kernel Loading Description.</td>
</tr>
<tr>
<td>Ascend-test_ops</td>
<td></td>
<td>Test operator package, which is used to test the computing power and power consumption of Ascend AI Processors.</td>
</tr>
<tr>
<td>Deep learning engine package</td>
<td>Ascend-fwkacllib</td>
<td>FwkACLlib supports offline inference, online inference, and training. Fwklib deliverables used to build graphs and operators online and load computational graphs to the NPU for execution, supporting training, offline model conversion, custom operator development, and fusion pattern development.</td>
</tr>
<tr>
<td>Ascend-cann-nnae_*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Package</td>
<td>Subpackage</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Ascend-opp</td>
<td></td>
<td>Operator library, including operator prototype libraries, operator implementation libraries, operator plugins, and fusion patterns. The operator implementation library includes TBE operators, AI CPU operators, and operator parsers.</td>
</tr>
<tr>
<td>Ascend-pyACL</td>
<td></td>
<td>pyACL installation package, including the Python library and sample code required for pyACL-based development.</td>
</tr>
<tr>
<td>aicpu_kernels</td>
<td></td>
<td>Operator package, providing the kernel implementation of AI CPU operators. For details about AI CPU kernel loading, see 13.3 AI CPU Kernel Loading Description.</td>
</tr>
<tr>
<td>Ascend-test_ops</td>
<td></td>
<td>Test operator package, which is used to test the computing power and power consumption of Ascend AI Processors.</td>
</tr>
<tr>
<td>Framework plugin package</td>
<td>Ascend-cann-tfplugin_*</td>
<td>Plugin package, an adaptation plugin used to connect to the upper-layer framework TensorFlow.</td>
</tr>
</tbody>
</table>
12 FAQ

12.1 What Do I Do If the "Software Has Been Installed" During RUN Package Installation?

12.2 "The user and group are not same with last installation" Is Displayed During Driver Installation

12.3 What Do I Do When the Error Message Indicating SO File Conflict Occurs Is Displayed During openssl-devel Installation?

12.4 What Do I Do When the Error Message "read time out" is Displayed When pip3.7 is Run?

12.5 pip3.7 install scipy Error

12.6 pip3.7 install numpy Error

12.7 Error Message "subprocess.CalledProcessError: Command '('lsb_release', '-a')' return non-zero exit status 1" Is Displayed During pip3.7 Installation

12.8 Error Caused by Inconsistent Python Versions

12.9 How Do I Handle a .deb Package Installation Failure?

12.1 What Do I Do If the "Software Has Been Installed" During RUN Package Installation?

Symptom

The following message is displayed during installation:

```
run package is already installed, install failed
```

Solution

The software package cannot be installed repeatedly. You need to uninstall the software package and then install it. For details, see 9 Uninstallation.
12.2 "The user and group are not same with last installation" Is Displayed During Driver Installation

**Symptom**

The following information is displayed during driver installation:

```
The user and group are not same with last installation, do not support overwriting installation
```

**Possible Cause**

The specified driver running user is inconsistent with that recorded in the `/etc/ascend_install.info` file. As a result, the verification fails and the installation exits. The AI CPU is not uninstalled when the driver and CANN software are uninstalled. As a result, the `/etc/ascend_install.info` file is not cleared.

**Solution**

Uninstall the AI CPU and then install the driver. To uninstall the AI CPU, perform the following steps:

**Step 1** Log in to the installation environment as the `root` user.

**Step 2** Run the following command to access the directory where the uninstallation script is located:

```
cd /usr/local/Ascend/opp/aicpu/script
```

**Step 3** Run the `./uninstall.sh` command to run the script.

----End

12.3 What Do I Do When the Error Message Indicating SO File Conflict Occurs Is Displayed During openssl-devel Installation?

**Symptom**

When you run the `yum install -y openssl-devel` command to install openssl-devel, an error message indicating that SO file conflict occurs is displayed, as shown in the following figure.
Possible Cause

The native package openssl-libs-1.1.1f-7.h1.eulerosv2r9 conflicts with the openssl-SMx-libs-1.1.1f-7.eulerosv2r9 package to be installed.

Note: Do not forcibly uninstall openssl-libs because services with SSL authentication depend on this package. For example, the SCP or SSH service depends on the libcrypto.so.1.1 library of this package. If this package is lost, the RPM package will become invalid.

Solution

Step 1  Create a folder and go to the folder.
mkdir openDown
cd openDown

Step 2  Run the following command to download the OpenSSL RPM package to the current folder:
yum install --downloadonly --downloaddir=. openssl-devel

Step 3  Run the rpm command for forcible installation.
rpm -Uvh *.rpm --force

12.4 What Do I Do When the Error Message "read time out" is Displayed When pip3.7 is Run?

Symptom

An error is reported when you use pip 3.7 to download related dependencies. For example, when you run the pip3.7 install scipy command to install the dependency, the error message "read time out" is displayed, as shown in the following figure.
Possible Cause

An unstable network prolongs the download time and slows down the download speed. This problem often occurs when large packages such as NumPy and SciPy are downloaded in socket communication.

Solution

Use the `-i` parameter to specify other common sources, as shown in the following command:

```
$ pip3.7 install scipy -i xxxxxx
```

`xxxxxx` indicates the available source entered based on your requirements.

12.5 pip3.7 install scipy Error

Symptom

The following error message is displayed during SciPy installation.
Possible Cause

The error message indicates that the lapack and blas libraries cannot be found in the /usr/lib64 directory. Actually, the libraries exist in the directory. It is the so.3 library that is unidentified.

```
root@localhost usr# find -name liblapack*
/lib64/liblapack.so.3.8
/lib64/liblapack.so.3.8.0
/lib64/liblapack.so.3
/lib64/liblapack.so.3.8
```

```
root@localhost usr# find -name libblas*
/lib64/libblas.so.3.8
/lib64/libblas.so.3.8.0
/lib64/libblas.so.3
/lib64/libblas.so.3.8
```

Solution

Create soft links to the `liblapack.so.3` and `libblas.so.3` files.

Run the following command as the installation user of the software package:

```
cd /usr/lib64
ln -s libblas.so.3 libblas.so
ln -s liblapack.so.3 liblapack.so
```

12.6 pip3.7 install numpy Error

Symptom

When you run the `pip3.7 install numpy` command to install the dependency, the error message "Could not build wheels for numpy which use PEP 517 and cannot be install directly" is displayed, as shown in the following figure.
Possible Cause

The default GCC version installed in the CentOS system is too early. As a result, the NumPy fails to be installed.

Solution

Run the following commands:

```
export CFLAGS=-std=c99
pip3.7 install numpy==1.17.2
```

12.7 Error Message "subprocess.CalledProcessError: Command '('lsb_release', '-a')' return non-zero exit status 1" Is Displayed During pip3.7 Installation

Symptom

During dependency installation, the error message "subprocess.CalledProcessError: Command '('lsb_release', '-a')' return non-zero exit status 1" is displayed when you run the `pip3.7 install xxx` command to install related software. The error message is as follows:
Possible Cause

When the subprocess module of Python 3.7.5 is executed, the system displays a message indicating that the lsb_release.py module cannot be found when the lsb_release -a command is executed. The lib path of Python 3.7.5 is /usr/local/python3.7.5/lib/python3.7/. The lsb_release.py module does not exist in the path. Therefore, an error is reported.

Solution

Step 1  Run the following command to search for the missing file lsb_release.py:

find / -name lsb_release

After the preceding command is executed, the following path is obtained. The path is only an example and may vary according to the actual situation.

/usr/bin/lsb_release

Step 2  Run the following command to back up the /usr/bin/lsb_release file in Step 1:

mv /usr/bin/lsb_release /usr/bin/lsb_release.bak

Step 3  Run the pip3.7.5 list command to check whether the fault is rectified.

---End

12.8 Error Caused by Inconsistent Python Versions

Symptom

For a common user, if the Python version on which other software depends is inconsistent with the required Python version, an error may occur.

Solution

You are advised to use the venv of Python. The Python virtual environment is used to isolate software package installation from the system.

12.9 How Do I Handle a .deb Package Installation Failure?

Symptom

If the .deb package fails to be installed, the apt command cannot be properly used.

Solution

If the .deb package fails to be installed, you need to manually delete the cache information so that the apt command can be used properly. The operation procedure is as follows:

Step 1  Run the following command to delete the cache information (the NNAE package is used as an example):
rm /var/lib/dpkg/info/cann-nnae*

Step 2  Run the following command to forcibly delete the file:

dpkg --remove --force-remove-reinstreq cann-nnae

Step 3  Delete the residual files in the directory of the software that fails to be installed.

rm -rf /usr/local/Ascend/nnae

----End
13 References

13.1 Parameters

One-click installation is supported in the command line. You can select parameters as required to complete the installation. All parameters are optional.

Installation command format: ./*.run [options]

For details, see Table 13-1.

NOTICE

If the parameters queried by running the ./*.run --help command are not described in the following table, this parameter is reserved or applies to other chip versions. You do not need to pay attention to this parameter.

Table 13-1 Parameters supported by the installation package

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--help</td>
<td>Queries help information.</td>
</tr>
<tr>
<td>--version</td>
<td>Queries version information.</td>
</tr>
<tr>
<td>--info</td>
<td>Queries software package construction information.</td>
</tr>
<tr>
<td>--list</td>
<td>Queries the software package list.</td>
</tr>
<tr>
<td>--check</td>
<td>Checks the consistency and integrity of software packages.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>--quiet</td>
<td>Silent installation, skipping interactive messages.</td>
</tr>
<tr>
<td>--noexec</td>
<td>Decompresses a software package to the current directory without running the installation script. This parameter is used together with --extract=&lt;path&gt;. The format is as follows: --noexec --extract=&lt;path&gt;</td>
</tr>
<tr>
<td>--extract=&lt;path&gt;</td>
<td>Decompresses a software package to a specified directory.</td>
</tr>
<tr>
<td>--tar [arg1 [arg2 ...]]</td>
<td>Runs the tar command on the software package. Use the arguments following tar as the command arguments. For example, the --tar xvf command indicates that the .run package will be decompressed to the current directory.</td>
</tr>
<tr>
<td>--install</td>
<td>Installs a software package. You can specify the installation path --install-path=&lt;path&gt; or use the default installation path.</td>
</tr>
<tr>
<td>--install-for-all</td>
<td>Allows all users to have the same installation group permission. If this option is included in installation or upgrade command, all users have the same permission on the directories and files created by the runfile installer as the installation group. This parameter must be used together with one of the parameters --install, --devel, and --upgrade, for example, /*/.run --install --install-for-all.</td>
</tr>
<tr>
<td>--install-username=&lt;username&gt;</td>
<td>Initial installation: You can specify the running user name. Otherwise, HwHiAiUser is used by default. Overwrite: The user name used in the last installation is adopted. This parameter must be used together with --install-usergroup=&lt;usergroup&gt;.</td>
</tr>
<tr>
<td></td>
<td>NOTE</td>
</tr>
<tr>
<td></td>
<td>Make sure the security risks are considered before you include this option.</td>
</tr>
<tr>
<td></td>
<td>NOTE</td>
</tr>
<tr>
<td></td>
<td>You are not advised to specify the root user for security considerations. To specify the root user, --install-for-all option should be used in pair (security risks exist).</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>--install-usergroup=&lt;usergroup&gt;</td>
<td>Initial installation: You can specify the running user group name. Otherwise, HwHiAiUser is used by default. Overwrite: The user group name used in the last installation is adopted. This parameter must be used together with --install-username=&lt;username&gt;.</td>
</tr>
</tbody>
</table>
| --install-path=<path> | Specifies the installation path. If the global configuration file ascend_cann_install.info exists in the environment, this parameter can be used, but the specified path must be the same as the installation path in the global configuration file. If you want to change the installation path, uninstall the CANN software package in the original path and ensure that the global configuration file ascend_cann_install.info has been deleted. You can check whether the file exists in the following directory:  
  ● root user: /etc/Ascend  
  ● Non-root user: $HOME/Ascend  
If you do not specify the path, the default path is used.  
  ● For installation as user root, the default installation path is /usr/local/Ascend.  
  ● For installation as a non-root user, the default installation path is $HOME/Ascend.  
If this parameter is used to specify the installation directory, the running user must have the read and write permissions on the specified installation directory. |
<p>| --uninstall | Uninstalls the software that has been installed. |
| --upgrade | Upgrades the software that has been installed. The system automatically checks the version number. If the version number is not in ascending order, the upgrade cannot be performed. |
| --devel | Installs a software package in development mode, that is, install only the files required by the development environment. |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| `--chip=<chip_type>` | Specifies the processor model so that the matching software (such as the AICPU operator package) can be selected during the installation. The options of `chip_type` are as follows:  
  - Ascend310: Ascend 310 PCIe processor.  
  - Ascend910: Ascend 910 PCIe processor.  
  - Ascend310-minirc: Ascend 310 SoC (It is the same as the Ascend 310 Processor, but it is started in RC mode and functions as the CPU of the main control board.) |
| NOTICE            | This parameter is valid only for the Ascend-cann-toolkit, Ascend-cann-nnrt and Ascend-cann-nnae software packages. If this parameter is not specified, all AI CPU operator packages in the software package are installed by default. |
| `--alternative=<feature1, feature2, feature3,...>` | Only the Toolkit software package supports this parameter.  
  Used to query the list of optional installation features. This parameter is used in optional installation scenarios. Before the installation, query the list of optional installation features supported by the software package so that the list can be copied to the input parameters `--blacklist` and `--whitelist`.  
  - The optional installation features for the Toolkit are as follows:  
    - atc: supports model compilation and conversion.  
    - devtools: debugging and simulation tool. If this feature is selected, nnae must be installed.  
    - nnrt: supports offline inference.  
    - nnae: supports offline inference, online inference, training, and IR graph building. |
| `--blacklist=<feature1, feature2, feature3,...>` | Optional installation blacklist, which is used to block some optional installation features during the installation (`--install`). This parameter cannot be used in upgrade scenarios.  
  This parameter must be used together with `--install`.  
  NOTE This parameter cannot be used together with `--whitelist`. |
### Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>--whitelist=&lt;feature1, feature2, feature3,...&gt;</code></td>
<td>Optional installation trustlist, which is used to specify some optional installation features during the installation (<code>--install</code>). This parameter cannot be used in upgrade scenarios. This parameter must be used together with <code>--install</code>. <strong>NOTE</strong> This parameter cannot be used together with <code>--blacklist</code>.</td>
</tr>
</tbody>
</table>

#### 13.2 File Paths

During the installation of the CANN software package, related configurations and log information are generated. **Table 13-2** lists the paths for storing these files. 

*{arch}* indicates the CPU architecture, and *${HOME}$* indicates the directory of the current user.

**Table 13-2** File paths

<table>
<thead>
<tr>
<th>Name</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software package installation log path</td>
<td><em>root</em> user: <code>/var/log/ascend_seclog/ascend_toolkit_install.log</code>&lt;br&gt;<code>Non-root user: </code>${HOME}/var/log/ascend_seclog/ascend_toolkit_install.log`</td>
</tr>
<tr>
<td>Path for recording information such as the software package version, CPU architecture, and installation path after the installation</td>
<td><em>The following uses the default installation path of the Toolkit software package as an example. Replace it as required.</em>&lt;br&gt;<code>root user: </code>/usr/local/Ascend/ascend-toolkit/latest/{arch}-linux/ascend_toolkit_install.info<code>&lt;br&gt;</code>Non-root user: <code>${HOME}/Ascend/ascend-toolkit/latest/{arch}-linux/ascend_toolkit_install.info</code></td>
</tr>
<tr>
<td>Path for recording the installation paths of software packages</td>
<td>*root user: <code>/etc/Ascend/ascend_cann_install.info</code>&lt;br&gt;<code>Non-root user: </code>${HOME}/Ascend/ascend_cann_install.info`</td>
</tr>
<tr>
<td>Path for recording installation parameters (such as <code>--install-for-all</code> and <code>--whitelist</code>) specified during software package installation</td>
<td><em>The following uses the default installation path of the Toolkit software package as an example. Replace it as required.</em>&lt;br&gt;<code>root user: </code>/usr/local/Ascend/ascend-toolkit/latest/{arch}-linux/install.conf<code>&lt;br&gt;</code>Non-root user: <code>${HOME}/Ascend/ascend-toolkit/latest/{arch}-linux/install.conf</code></td>
</tr>
</tbody>
</table>
13.3 AI CPU Kernel Loading Description

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Description</th>
<th>Restrictions</th>
</tr>
</thead>
</table>
| PM       | After the AI CPU kernel is installed, it is initialized with the service process and automatically loaded to the device. | ● When a Da Vinci Mini chip is used by multiple users at the same time, the users cannot load the AI CPU kernels of different versions.  
● A Da Vinci Mini chip cannot be specified to load the AI CPU kernel of a specific version in the operating environment. |

13.4 Product Modes in Inference Scenario

The following uses the Ascend AI Processor as an example. If the PCIe works in master mode and supports peripherals, this is referred to as RC mode (or Ascend RC in the following). If the PCIe works in slave mode, this is referred to as EP mode (or Ascend EP in the following).

- The working modes of the Ascend AI Processor are as follows:
  - The Ascend 310 AI Processor supports both the EP and RC modes.
  - The Ascend 910 AI Processor supports only the EP mode.
- The following products support RC mode: Atlas 200 AI accelerator module and Atlas 200 DK developer kit. The CPUs of such products run the AI service software specified by the running user directly and connect to peripherals such as network cameras, I²C sensors, and SPI monitors as slave devices.
- The following products support EP mode:
  - Ascend 910 AI Processor: Atlas 800 training server and Atlas 300T training card
In EP mode, the host acts as the master, the device acts as the slave, and customer AI service programs run on the host. The product functions as a device system and connects to the host system through the PCIe channel, while the host interacts with the device system through the PCIe channel and loads AI tasks to the Ascend AI Processor on the device.
Figure 13-1 shows the products and architecture of the two modes.

- The host is an x86 server, an ARM server, or a Windows PC connected to hardware equipped with an Ascend AI Processor. The host leverages the neural network (NN) compute capability provided by the Ascend AI Processor to implement services.
- A device is a hardware backend equipped with an Ascend AI Processor. It is connected to the server over the PCIe interface and provides the NN compute capability required by the server.

Figure 13-1 RC and EP modes