ROCm installation on Linux

Release 6.4.1

Advanced Micro Devices, Inc.

Mar 31, 2026

INSTALL ROCM

This section describes the ROCm for Linux installation options.

> **Note**
>
> If you're using ROCm with AMD Radeon or Radeon Pro GPUs for graphics workloads, see the Use ROCm on Radeon GPU documentation for installation instructions.

Install ROCm

- Quick start - recommended for new users
- Detailed install - includes explanations

Install deep learning frameworks

- PyTorch
- TensorFlow
- JAX
- DGL

The documentation is structured as follows:

How to

- Run Docker containers
- Use Spack

Reference

- Package manager integration
- System requirements (Linux)
- Troubleshooting
- User and kernel-space support matrix

CHAPTER

ONE

QUICK START INSTALLATION GUIDE

This topic provides basic installation instructions for ROCm on Linux using your distribution's native package manager. Before you begin, you should confirm your kernel version matches the ROCm system requirements.

Once you do, review your required installation instructions by selecting your operating system and version, and then run the provided commands in your terminal. The commands include the installation of the prerequisites, along with installing ROCm.

For more in-depth installation instructions, refer to ROCm on Linux detailed installation overview.

> **Note**
>
> If you're using ROCm with AMD Radeon or Radeon Pro GPUs for graphics workloads, see the Use ROCm on Radeon GPU documentation for installation instructions .

1.1 ROCm installation

Ubuntu

24.04

```
wget https://repo.radeon.com/amdgpu-install/6.4.1/ubuntu/noble/amdgpu-install_6.4.60401-1_all.deb
sudo apt install ./amdgpu-install_6.4.60401-1_all.deb
sudo apt update
sudo apt install python3-setuptools python3-wheel
sudo usermod -a -G render,video $LOGNAME # Add the current user to the render and video groups
sudo apt install rocm
```

22.04

```
wget https://repo.radeon.com/amdgpu-install/6.4.1/ubuntu/jammy/amdgpu-install_6.4.60401-1_all.deb
sudo apt install ./amdgpu-install_6.4.60401-1_all.deb
sudo apt update
sudo apt install python3-setuptools python3-wheel
sudo usermod -a -G render,video $LOGNAME # Add the current user to the render and video groups
sudo apt install rocm
```

Debian

12

```
wget https://repo.radeon.com/amdgpu-install/6.4.1/ubuntu/jammy/amdgpu-install_6.4.60401-1_all.deb
sudo apt install ./amdgpu-install_6.4.60401-1_all.deb
sudo apt update
sudo apt install -y python3-setuptools python3-wheel
sudo usermod -a -G render,video $LOGNAME # Add the current user to the render and video groups
sudo apt install rocm
```

Red Hat Enterprise Linux

9.6

Before installing ROCm on RHEL, register your Enterprise Linux and update the OS installation.

```
sudo dnf install https://repo.radeon.com/amdgpu-install/6.4.1/rhel/9.6/amdgpu-install-6.4.60401-1.el9.
↪noarch.rpm
sudo dnf clean all
wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.rpm
sudo rpm -ivh epel-release-latest-9.noarch.rpm
sudo dnf install dnf-plugin-config-manager
sudo crb enable
sudo dnf install python3-setuptools python3-wheel
sudo usermod -a -G render,video $LOGNAME # Add the current user to the render and video groups
sudo dnf install rocm
```

9.5

Before installing ROCm on RHEL, register your Enterprise Linux and update the OS installation.

```
sudo dnf install https://repo.radeon.com/amdgpu-install/6.4.1/rhel/9.5/amdgpu-install-6.4.60401-1.el9.
↪noarch.rpm
sudo dnf clean all
wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.rpm
sudo rpm -ivh epel-release-latest-9.noarch.rpm
sudo dnf install dnf-plugin-config-manager
sudo crb enable
sudo dnf install python3-setuptools python3-wheel
sudo usermod -a -G render,video $LOGNAME # Add the current user to the render and video groups
sudo dnf install rocm
```

9.4

Before installing ROCm on RHEL, register your Enterprise Linux and update the OS installation.

```
sudo dnf install https://repo.radeon.com/amdgpu-install/6.4.1/rhel/9.4/amdgpu-install-6.4.60401-1.el9.
↪noarch.rpm
sudo dnf clean all
wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.rpm
sudo rpm -ivh epel-release-latest-9.noarch.rpm
sudo dnf install dnf-plugin-config-manager
sudo crb enable
```

(continues on next page)

(continued from previous page)

```
sudo dnf install python3-setuptools python3-wheel
sudo usermod -a -G render,video $LOGNAME # Add the current user to the render and video groups
sudo dnf install rocm
```

8.10

Before installing ROCm on RHEL, register your Enterprise Linux and update the OS installation.

```
sudo dnf install https://repo.radeon.com/amdgpu-install/6.4.1/rhel/8.10/amdgpu-install-6.4.60401-1.el8.
↪noarch.rpm
sudo dnf clean all
wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
sudo rpm -ivh epel-release-latest-8.noarch.rpm
sudo dnf install dnf-plugin-config-manager
sudo crb enable
sudo dnf install python3-setuptools python3-wheel
sudo usermod -a -G render,video $LOGNAME # Add the current user to the render and video groups
sudo dnf install rocm
```

Oracle Linux

9

Before installing ROCm on OL, update the OS installation.

```
sudo dnf install https://repo.radeon.com/amdgpu-install/6.4.1/el/9.5/amdgpu-install-6.4.60401-1.el9.
↪noarch.rpm
sudo dnf clean all
wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.rpm
sudo rpm -ivh epel-release-latest-9.noarch.rpm
sudo dnf install dnf-plugin-config-manager
sudo crb enable
sudo dnf install python3-setuptools python3-wheel
sudo usermod -a -G render,video $LOGNAME # Add the current user to the render and video groups
sudo dnf install rocm
```

8

Before installing ROCm on OL, update the OS installation.

```
sudo dnf install https://repo.radeon.com/amdgpu-install/6.4.1/el/8.10/amdgpu-install-6.4.60401-1.el8.
↪noarch.rpm
sudo dnf clean all
wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
sudo rpm -ivh epel-release-latest-8.noarch.rpm
sudo dnf install dnf-plugin-config-manager
sudo crb enable
sudo dnf install python3-setuptools python3-wheel
sudo usermod -a -G render,video $LOGNAME # Add the current user to the render and video groups
sudo dnf install rocm
```

SUSE Linux Enterprise Server

15.6

Before installing ROCm on SLES, register your Enterprise Linux update the OS installation.

```
sudo SUSEConnect -p sle-module-desktop-applications/15.6/x86_64
sudo SUSEConnect -p sle-module-development-tools/15.6/x86_64
sudo SUSEConnect -p PackageHub/15.6/x86_64
sudo zypper install zypper
sudo zypper --no-gpg-checks install https://repo.radeon.com/amdgpu-install/6.4.1/sle/15.6/amdgpu-
↪install-6.4.60401-1.noarch.rpm
sudo zypper --gpg-auto-import-keys refresh
sudo zypper addrepo https://download.opensuse.org/repositories/devel:languages:perl/15.6/
↪devel:languages:perl.repo
sudo zypper addrepo https://download.opensuse.org/repositories/Education/15.6/Education.repo
sudo zypper addrepo https://download.opensuse.org/repositories/science/SLE_15_SP5/science.repo
sudo zypper --gpg-auto-import-keys refresh
sudo zypper install python3-setuptools python3-wheel
sudo usermod -a -G render,video $LOGNAME # Add the current user to the render and video groups
sudo zypper install rocm
```

Azure Linux

3.0

```
sudo tdnf install dnf-plugin-config-manager
sudo curl -o /etc/yum.repos.d/azurelinux-extended.repo https://packages.microsoft.com/azurelinux/3.0/
↪prod/extended/x86_64/config.repo
sudo tdnf install python3-setuptools python3-wheel
sudo usermod -a -G render,video $LOGNAME # Add the current user to the render and video groups
sudo tee --append /etc/yum.repos.d/rocm.repo <<EOF
[ROCm-6.4.1]
name=ROCm6.4.1
baseurl=https://repo.radeon.com/rocm/azurelinux3/6.4.1/main/
enabled=1
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
sudo tdnf install rocm
sudo tdnf clean all
```

1.2 AMDGPU driver installation

Ubuntu

24.04

```
wget https://repo.radeon.com/amdgpu-install/6.4.1/ubuntu/noble/amdgpu-install_6.4.60401-1_all.deb
sudo apt install ./amdgpu-install_6.4.60401-1_all.deb
sudo apt update
sudo apt install "linux-headers-$(uname -r)" "linux-modules-extra-$(uname -r)"
sudo apt install amdgpu-dkms
```

22.04

```
wget https://repo.radeon.com/amdgpu-install/6.4.1/ubuntu/jammy/amdgpu-install_6.4.60401-1_all.deb
sudo apt install ./amdgpu-install_6.4.60401-1_all.deb
sudo apt update
sudo apt install "linux-headers-$(uname -r)" "linux-modules-extra-$(uname -r)"
sudo apt install amdgpu-dkms
```

Debian

12

```
wget https://repo.radeon.com/amdgpu-install/6.4.1/ubuntu/jammy/amdgpu-install_6.4.60401-1_all.deb
sudo apt install ./amdgpu-install_6.4.60401-1_all.deb
sudo apt update
sudo apt install "linux-headers-$(uname -r)"
sudo apt install amdgpu-dkms
```

Red Hat Enterprise Linux

9.6

Before installing AMDGPU driver on RHEL, register your Enterprise Linux and update the OS installation.

```
sudo dnf install https://repo.radeon.com/amdgpu-install/6.4.1/rhel/9.6/amdgpu-install-6.4.60401-1.el9.
↪noarch.rpm
sudo dnf clean all
sudo dnf install "kernel-headers-$(uname -r)" "kernel-devel-$(uname -r)" "kernel-devel-matched-$(uname -
↪r)"
sudo dnf install amdgpu-dkms
```

9.5

Before installing AMDGPU driver on RHEL, register your Enterprise Linux and update the OS installation.

```
sudo dnf install https://repo.radeon.com/amdgpu-install/6.4.1/rhel/9.5/amdgpu-install-6.4.60401-1.el9.
↪noarch.rpm
sudo dnf clean all
sudo dnf install "kernel-headers-$(uname -r)" "kernel-devel-$(uname -r)" "kernel-devel-matched-$(uname -
↪r)"
sudo dnf install amdgpu-dkms
```

9.4

Before installing AMDGPU driver on RHEL, register your Enterprise Linux and update the OS installation.

```
sudo dnf install https://repo.radeon.com/amdgpu-install/6.4.1/rhel/9.4/amdgpu-install-6.4.60401-1.el9.
↪noarch.rpm
sudo dnf clean all
sudo dnf install "kernel-headers-$(uname -r)" "kernel-devel-$(uname -r)" "kernel-devel-matched-$(uname -
↪r)"
sudo dnf install amdgpu-dkms
```

8.10

Before installing AMDGPU driver on RHEL, register your Enterprise Linux and update the OS installation.

```
sudo dnf install https://repo.radeon.com/amdgpu-install/6.4.1/rhel/8.10/amdgpu-install-6.4.60401-1.el8.
↪noarch.rpm
sudo dnf clean all
sudo dnf install "kernel-headers-$(uname -r)" "kernel-devel-$(uname -r)"
sudo dnf install amdgpu-dkms
```

Oracle Linux

9

Before installing AMDGPU driver on OL, update the OS installation.

```
sudo dnf install https://repo.radeon.com/amdgpu-install/6.4.1/el/9.5/amdgpu-install-6.4.60401-1.el9.
↪noarch.rpm
sudo dnf clean all
sudo dnf install "kernel-uek-devel-$(uname -r)"
sudo dnf install amdgpu-dkms
```

8

Before installing AMDGPU driver on OL, update the OS installation.

```
sudo dnf install https://repo.radeon.com/amdgpu-install/6.4.1/el/8.10/amdgpu-install-6.4.60401-1.el8.
↪noarch.rpm
sudo dnf clean all
sudo dnf install "kernel-uek-devel-$(uname -r)"
sudo dnf install amdgpu-dkms
```

SUSE Linux Enterprise Server

15.6

Before installing AMDGPU driver on SLES, register your Enterprise Linux and update the OS installation.

```
sudo SUSEConnect -p sle-module-desktop-applications/15.6/x86_64
sudo SUSEConnect -p sle-module-development-tools/15.6/x86_64
sudo SUSEConnect -p PackageHub/15.6/x86_64
sudo zypper install zypper
sudo zypper --no-gpg-checks install https://repo.radeon.com/amdgpu-install/6.4.1/sle/15.6/amdgpu-
↪install-6.4.60401-1.noarch.rpm
sudo zypper --gpg-auto-import-keys refresh
sudo zypper install kernel-default-devel
sudo zypper install amdgpu-dkms
```

Azure Linux

3.0

```
sudo tdnf install "kernel-headers-$(uname -r)" "kernel-devel-$(uname -r)"
sudo tdnf install azurelinux-repos-amd
```

(continues on next page)

(continued from previous page)

```
sudo tdnf repolist --refresh
sudo tdnf install amdgpu
```

> **Note**
>
> For Azure Linux, the latest amdgpu version supported is from the ROCm 6.2.2 release.

> **Important**
>
> To apply all settings, reboot your system.

> **Note**
>
> Quick Start enables GPU access for the current user only. To grant GPU access to all users, see Configuring permissions for GPU access.

After completing the installation, review the Post-installation instructions. If you have issues with your installation, see Troubleshooting.

CHAPTER

TWO

ROCM ON LINUX DETAILED INSTALLATION OVERVIEW

To install ROCm, please follow these steps

1. Install and confirm prerequisites
2. Choose installation method
3. Complete post-installation steps

If you have a problem with your ROCm on Linux installation, see installation troubleshooting. Also, consider sharing the problem in the ROCm on Linux Github repository in the issues section.

2.1 Installation prerequisites

Before installing ROCm, complete the following prerequisites.

1. Confirm the system has a supported Linux version.
   - To obtain the Linux distribution information, use the following command:

     ```
     uname -m && cat /etc/*release
     ```

   - Confirm that your Linux distribution matches a supported distribution.

     Example: Running the preceding command on an Ubuntu system produces the following output:

     ```
     x86_64
     DISTRIB_ID=Ubuntu
     DISTRIB_RELEASE=20.04
     DISTRIB_CODENAME=focal
     DISTRIB_DESCRIPTION="Ubuntu 20.04.5 LTS"
     ```

2. Verify the kernel version.
   - To check the kernel version of your Linux system, type the following command:

     ```
     uname -srmv
     ```

     Example: The preceding command lists the kernel version in the following format:

     ```
     Linux 5.15.0-46-generic #44~20.04.5-Ubuntu SMP Fri Jun 24 13:27:29 UTC 2022 x86_64
     ```

   - Confirm that your kernel version matches the system requirements, as listed in Supported operating systems.

2.1.1 Register your Enterprise Linux

If you're using Red Hat Enterprise Linux (RHEL) or SUSE Linux Enterprise Server (SLES), register your operating system to ensure you're able to download and install packages.

Ubuntu

There is no registration required for Ubuntu.

Debian

There is no registration required for Debian.

Red Hat Enterprise Linux

Typically you can register by following the step-by-step user interface. If you need to register by command line, use the following commands:

```
subscription-manager register --username <username> --password <password>
subscription-manager attach --auto
```

More details about registering for RHEL

Oracle Linux

There is no registration required for Oracle Linux.

SUSE Linux Enterprise Server

Typically you can register by following the step-by-step user interface. If you need to register by command line, use the following commands:

```
sudo SUSEConnect -r <REGCODE>
```

More details about registering for SLES

Azure Linux

There is no registration required for Azure Linux.

2.1.2 Update your Enterprise Linux

If you are using Red Hat Enterprise Linux (RHEL) or SUSE Linux Enterprise Servers (SLES), or Oracle Linux, it is recommended that you update your operating system to the latest packages from the Linux distribution. This is a requirement for newer hardware on older versions of RHEL, SLES or OL.

Ubuntu

There is no update required for Ubuntu.

Debian

There is no update required for Debian.

Red Hat Enterprise Linux

9.6

```
sudo dnf update --releasever=9.6 --exclude=\*release\*
```

9.5

```
sudo dnf update --releasever=9.5 --exclude=\*release\*
```

9.4

```
sudo dnf update --releasever=9.4 --exclude=\*release\*
```

8.10

```
sudo dnf update --releasever=8.10 --exclude=\*release\*
```

Oracle Linux

9

```
sudo dnf update --releasever=9.5 --exclude=\*release\*
```

8

```
sudo dnf update --releasever=8.10 --exclude=\*release\*
```

SUSE Linux Enterprise Server

15.6

```
sudo zypper update
```

Azure Linux

There is no update required for Azure Linux.

Important

To apply all settings, reboot your system.

2.1.3 Additional package repositories

For some distributions, the ROCm installation packages depend on packages that aren't included in the default package repositories. These external repositories need to be sourced before installation. Use the following instructions specific to your distribution to add the necessary repositories.

Ubuntu

All ROCm installation packages are available in the default Ubuntu repositories.

Debian

All ROCm installation packages are available in the default Debian repositories.

Red Hat Enterprise Linux

1. Add the EPEL repository.

   ##### RHEL 9

   ```
   wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.rpm
   sudo rpm -ivh epel-release-latest-9.noarch.rpm
   ```

   ##### RHEL 8

   ```
   wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
   sudo rpm -ivh epel-release-latest-8.noarch.rpm
   ```

2. Enable the CodeReady Linux Builder (CRB) repository.

   In order to enable CRB, you may need to install dnf-plugin-config-manager first.

   ```
   sudo dnf install dnf-plugin-config-manager
   sudo crb enable
   ```

Oracle Linux

1. Add the EPEL repository.

   ##### 9

   ```
   wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.rpm
   sudo rpm -ivh epel-release-latest-9.noarch.rpm
   ```

   ##### 8

   ```
   wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
   sudo rpm -ivh epel-release-latest-8.noarch.rpm
   ```

2. Enable the CodeReady Linux Builder (CRB) repository.

   In order to enable CRB, you may need to install dnf-plugin-config-manager first.

   ```
   sudo dnf install dnf-plugin-config-manager
   sudo crb enable
   ```

SUSE Linux Enterprise Server

Add a few modules with SUSEConnect, along with the Perl language, Education and science repositories.

SLES 15.6

```
sudo SUSEConnect -p sle-module-desktop-applications/15.6/x86_64
sudo SUSEConnect -p sle-module-development-tools/15.6/x86_64
sudo SUSEConnect -p PackageHub/15.6/x86_64
sudo zypper install zypper
sudo zypper addrepo https://download.opensuse.org/repositories/devel:/languages:/perl/15.6/
↪devel:languages:perl.repo
sudo zypper addrepo https://download.opensuse.org/repositories/Education/15.6/Education.repo
sudo zypper addrepo https://download.opensuse.org/repositories/science/SLE_15_SP5/science.repo
```

Azure Linux

Enable the the config repository for additional packages. In order to enable config, you may need to install dnf-plugin-config-manager first.

AZL 3.0

```
sudo tdnf install dnf-plugin-config-manager
sudo curl -o /etc/yum.repos.d/azurelinux-extended.repo https://packages.microsoft.com/azurelinux/3.0/
↪prod/extended/x86_64/config.repo
```

2.1.4 Additional development packages

ROCm installation requires additional packages for operation and development.

To install the required packages, use the following instructions specific to your distribution:

Ubuntu

```
sudo apt install python3-setuptools python3-wheel
```

Debian

```
sudo apt install python3-setuptools python3-wheel
```

Red Hat Enterprise Linux

```
sudo dnf install python3-setuptools python3-wheel
```

Oracle Linux

```
sudo dnf install python3-setuptools python3-wheel
```

SUSE Linux Enterprise Server

```
sudo zypper install python3-setuptools python3-wheel
```

Azure Linux

```
sudo tdnf install python3-setuptools python3-wheel
```

Optionally, if configuring the post-ROCm installation using environment-modules, install the following:

Ubuntu

```
sudo apt install environment-modules
```

Debian

```
sudo apt install environment-modules
```

Red Hat Enterprise Linux

```
sudo dnf install environment-modules
```

Oracle Linux

```
sudo dnf install environment-modules
```

SUSE Linux Enterprise Server

```
sudo zypper install environment-modules

# Create a link for installed modules version
version=$(rpm -qa | grep '^Modules-' | awk -F'-' '{print $2}')
sudo ln -s /usr/share/Modules/$version/modulefiles /usr/share/Modules/modulefiles
```

Azure Linux

```
sudo tdnf install environment-modules
```

2.1.5 Configuring permissions for GPU access

There are two primary methods to configure GPU access for ROCm: group membership or udev rules. Each method has its own advantages, and the choice depends on your specific requirements and system management preferences.

2.1.5.1 Using group membership

By default, GPU access is managed through membership in the video and render groups. The video and render groups are system groups in Linux used to manage access to graphics hardware and related functionality. Traditionally, the video group is used to control access to video devices, including graphics cards and video capture devices. The render group is more recent and specifically controls access to GPU rendering capabilities through Direct Rendering Manager (DRM) render nodes.

1. To check the groups in your system, issue the following command:

   ```
   groups
   ```

2. Add yourself to the video and render groups:

```
sudo usermod -a -G video,render $LOGNAME
```

3. Optionally, add other users to the video and render groups:

```
sudo usermod -a -G video,render user1
sudo usermod -a -G video,render user2
```

4. To add all future users to the render and video groups by default, run the following commands:

```
echo 'ADD_EXTRA_GROUPS=1' | sudo tee -a /etc/adduser.conf
echo 'EXTRA_GROUPS=video' | sudo tee -a /etc/adduser.conf
echo 'EXTRA_GROUPS=render' | sudo tee -a /etc/adduser.conf
```

2.1.5.2 Using udev rules

A flexible way to manage device permissions is to use udev rules. They apply system-wide, can be easily deployed via configuration management tools, and eliminate the need for user group management. This method provides more granular control over GPU access.

2.1.5.2.1 Grant GPU access to all users on the system

1. Create a new file /etc/udev/rules.d/70-amdgpu.rules with the following content:

```
KERNEL==”kfd”, MODE=”0666”
SUBSYSTEM==”drm”, KERNEL==”renderD*”, MODE=”0666”
```

2. Reload the udev rules:

```
sudo udevadm control --reload-rules && sudo udevadm trigger
```

This configuration grants all users read and write access to AMD GPU resources, including the AMD Kernel-mode GPU Driver (KMD) and Direct Rendering Manager (DRM) devices.

2.1.5.2.2 Grant GPU access to a custom group

1. Create a new group (e.g., devteam):

```
sudo groupadd devteam
```

2. Add users to the new group:

```
sudo usermod -a -G devteam dev1
sudo usermod -a -G devteam dev2
```

3. Create udev rules to assign GPU devices to this group:

   Create a file /etc/udev/rules.d/70-amdgpu.rules with:

```
KERNEL==”kfd”, GROUP=”devteam”, MODE=”0660”
SUBSYSTEM==”drm”, KERNEL==”renderD*”, GROUP=”devteam”, MODE=”0660”
```

4. Reload the udev rules:

```
sudo udevadm control --reload-rules && sudo udevadm trigger
```

This configuration grants all users in the devteam group read and write access to AMD GPU resources, including the AMD Kernel-mode GPU Driver (KMD) and Direct Rendering Manager (DRM) devices.

2.1.6 Disable integrated graphics (IGP)

ROCm doesn't currently support integrated graphics. If your system has an AMD IGP installed, disable it in the BIOS prior to using ROCm. If the driver can enumerate the IGP, the ROCm runtime might crash the system, even when omission was specified via HIP_VISIBLE_DEVICES.

2.1.7 Secure Boot

When installing the AMDGPU driver with Secure Boot enabled, you must sign amdgpu-dkms to prevent potential system loading issues. For more information, see Secure Boot Support. If you prefer not to sign the AMDGPU driver, you can disable Secure Boot from the BIOS settings instead.

2.2 ROCm installation overview

If you're new to ROCm, we recommend using the Quick start installation guide.

> **Note**
>
> If you're using a Radeon GPU with graphical applications, refer to the Radeon installation instructions.

To install ROCm, you can use the package manager or the AMDGPU installer. You can also opt for single-version or multi-version installation. These topics are described in detail in the following sections.

2.2.1 Installation methods

- Package manager versus AMDGPU installer
- Multi-version installation
- ROCm Offline Installer Creator
- ROCm Runfile Installer

2.2.1.1 Package manager versus AMDGPU installer

ROCm supports two methods for installation:

- Using the Linux distribution package manager
- Running the amdgpu-install script

There is no difference in the final installation between these two methods.

Using the distribution's package manager lets the user install, upgrade and uninstall using familiar commands and workflows. Third party ecosystem support is the same as your OS package manager.

The amdgpu-install script is a wrapper around the package manager. The same packages are installed by this script as the package manager system.

The installer automates the installation process for the AMDGPU and ROCm stack. It handles the complete installation process for ROCm, including setting up the repository, cleaning the system, updating, and

installing the desired drivers and meta-packages. Users who are less familiar with the package manager can choose this method for ROCm installation.

2.2.1.2 Multi-version installation

A multi-version ROCm installation handles situations where users need multiple versions of ROCm on the same machine for compatibility with different applications and hardware, testing, and other use cases. For instructions, see installing multiple ROCm versions.

2.2.1.3 ROCm Offline Installer Creator

The ROCm Offline Installer Creator creates an installation package for a preconfigured setup of ROCm, the AMDGPU driver, or a combination of the two on a target system without network or internet access. See ROCm Offline Installer Creator for instructions.

2.2.1.4 ROCm Runfile Installer

The ROCm Runfile Installer lets you install ROCm without using a native Linux package management system. It can be used with or without network or internet access. See ROCm Runfile Installer for instructions.

2.2.2 Installation via native package manager

Select the install instructions for your operating system

Install

- Ubuntu
- Debian
- Red Hat Enterprise Linux
- Oracle Linux
- SUSE Linux Enterprise Server
- Azure Linux

Uninstall

- Ubuntu
- Debian
- Red Hat Enterprise Linux
- Oracle Linux
- SUSE Linux Enterprise Server
- Azure Linux

See also: System requirements (Linux). If you encounter install issues, you can refer to the troubleshooting page.

2.2.2.1 Ubuntu native installation

> **Caution**
>
> Ensure that the Installation prerequisites are met before installing.

2.2.2.1.1 Registering ROCm repositories

2.2.2.1.1.1 Package signing key

Download and convert the package signing key.

```
# Make the directory if it doesn't exist yet.
# This location is recommended by the distribution maintainers.
sudo mkdir --parents --mode=0755 /etc/apt/keyrings

# Download the key, convert the signing-key to a full
# keyring required by apt and store in the keyring directory
wget https://repo.radeon.com/rocm/rocm.gpg.key -O - | \
    gpg --dearmor | sudo tee /etc/apt/keyrings/rocm.gpg > /dev/null
```

> **Note**
>
> The GPG key may change; ensure it is updated when installing a new release. If the key signature verification fails while updating, re-add the key from the ROCm to the apt repository as mentioned above.

2.2.2.1.1.2 Register packages

Ubuntu 24.04

```
# Register ROCm packages
echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/rocm/apt/6.4.
↪1 noble main" \
    | sudo tee --append /etc/apt/sources.list.d/rocm.list
echo -e 'Package: *\nPin: release o=repo.radeon.com\nPin-Priority: 600' \
    | sudo tee /etc/apt/preferences.d/rocm-pin-600
sudo apt update
```

Ubuntu 22.04

```
# Register ROCm packages
echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/rocm/apt/6.4.
↪1 jammy main" \
    | sudo tee --append /etc/apt/sources.list.d/rocm.list
echo -e 'Package: *\nPin: release o=repo.radeon.com\nPin-Priority: 600' \
    | sudo tee /etc/apt/preferences.d/rocm-pin-600
sudo apt update
```

2.2.2.1.2 Installing

2.2.2.1.2.1 Install kernel driver

For information about the AMDGPU driver installation, see the Ubuntu native installation in the AMD Instinct Data Center GPU Documentation.

For information about driver compatibility, see User and kernel-space support matrix.

2.2.2.1.2.2 Install ROCm

```
sudo apt install rocm
```

Complete the Post-installation instructions.

2.2.2.1.3 Upgrading

To upgrade an existing ROCm installation to a newer version, follow the steps in Registering ROCm repositories and Installing.

> **Note**
>
> Upgrading the kernel driver may also upgrade the GPU firmware, which requires a system reboot to take effect.

2.2.2.1.4 Uninstalling

2.2.2.1.4.1 Uninstall specific meta packages

```
sudo apt autoremove rocm
```

2.2.2.1.4.2 Uninstall ROCm packages

```
sudo apt autoremove rocm-core
```

2.2.2.1.4.3 Remove ROCm repositories

```
# Remove the repositories
sudo rm /etc/apt/sources.list.d/rocm.list

# Clear the cache and clean the system
sudo rm -rf /var/cache/apt/*
sudo apt clean all
sudo apt update

# Restart the system
sudo reboot
```

2.2.2.2 Debian native installation

> **Caution**
>
> Ensure that the Installation prerequisites are met before installing.

2.2.2.2.1 Registering ROCm repositories

2.2.2.2.1.1 Package signing key

Download and convert the package signing key.

```
# Make the directory if it doesn't exist yet.
# This location is recommended by the distribution maintainers.
sudo mkdir --parents --mode=0755 /etc/apt/keyrings

# Download the key, convert the signing-key to a full
# keyring required by apt and store in the keyring directory
wget https://repo.radeon.com/rocm/rocm.gpg.key -O - | \
    gpg --dearmor | sudo tee /etc/apt/keyrings/rocm.gpg > /dev/null
```

> **Note**
>
> The GPG key may change; ensure it is updated when installing a new release. If the key signature verification fails while updating, re-add the key from the ROCm to the apt repository as mentioned above.

2.2.2.2.1.2 Register packages

Debian 12

```
# Register ROCm packages
echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/rocm/apt/6.4.
↪1 jammy main" \
    | sudo tee --append /etc/apt/sources.list.d/rocm.list
echo -e 'Package: *\nPin: release o=repo.radeon.com\nPin-Priority: 600' \
    | sudo tee /etc/apt/preferences.d/rocm-pin-600
sudo apt update
```

2.2.2.2.2 Installing

2.2.2.2.2.1 Install kernel driver

For information about the AMDGPU driver installation, see the Debian native installation in the AMD Instinct Data Center GPU Documentation.

For information about driver compatibility, see User and kernel-space support matrix.

2.2.2.2.2.2 Install ROCm

```
sudo apt install rocm
```

Complete the Post-installation instructions.

2.2.2.2.3 Upgrading

To upgrade an existing ROCm installation to a newer version, follow the steps in Registering ROCm repositories and Installing.

> **Note**
>
> Upgrading the kernel driver may also upgrade the GPU firmware, which requires a system reboot to take effect.

2.2.2.2.4 Uninstalling

2.2.2.2.4.1 Uninstall specific meta packages

```
sudo apt autoremove rocm
```

2.2.2.2.4.2 Uninstall ROCm packages

```
sudo apt autoremove rocm-core
```

2.2.2.2.4.3 Remove ROCm repositories

```
# Remove the repositories
sudo rm /etc/apt/sources.list.d/rocm.list

# Clear the cache and clean the system
sudo rm -rf /var/cache/apt/*
sudo apt clean all
sudo apt update

# Restart the system
sudo reboot
```

2.2.2.3 Red Hat Enterprise Linux native installation

> **Caution**
>
> Ensure that the Installation prerequisites are met before installing.

2.2.2.3.1 Registering ROCm repositories

RHEL 9

```
sudo tee --append /etc/yum.repos.d/rocm.repo <<EOF
[ROCm-6.4.1]
name=ROCm6.4.1
baseurl=https://repo.radeon.com/rocm/el9/6.4.1/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
sudo dnf clean all
```

RHEL 8

```
sudo tee --append /etc/yum.repos.d/rocm.repo <<EOF
[ROCm-6.4.1]
name=ROCm6.4.1
baseurl=https://repo.radeon.com/rocm/el8/6.4.1/main
```

(continues on next page)

(continued from previous page)

```
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
sudo dnf clean all
```

2.2.2.3.2 Installing

2.2.2.3.2.1 Install kernel driver

For information about the AMDGPU driver installation, see the Red Hat Enterprise Linux native installation in the AMD Instinct Data Center GPU Documentation.

For information about driver compatibility, see User and kernel-space support matrix.

2.2.2.3.2.2 Install ROCm

```
sudo dnf install rocm
```

Complete the Post-installation instructions.

2.2.2.3.3 Upgrading

To upgrade an existing ROCm installation to a newer version, follow the steps in Registering ROCm repositories and Installing.

> **Note**
>
> Upgrading the kernel driver may also upgrade the GPU firmware, which requires a system reboot to take effect.

2.2.2.3.4 Uninstalling

2.2.2.3.4.1 Uninstall specific meta packages

```
sudo dnf remove rocm
```

2.2.2.3.4.2 Uninstall ROCm packages

```
sudo dnf remove rocm-core amdgpu-core
```

2.2.2.3.4.3 Remove ROCm repositories

```
# Remove the repositories
sudo rm /etc/yum.repos.d/rocm.repo*

# Clear the cache and clean the system
sudo rm -rf /var/cache/dnf
```

(continues on next page)

(continued from previous page)

```
sudo dnf clean all

# Restart the system
sudo reboot
```

2.2.2.4 Oracle Linux native installation

> **Caution**
>
> Ensure that the Installation prerequisites are met before installing.

2.2.2.4.1 Register ROCm repositories

OL 9

```
sudo tee --append /etc/yum.repos.d/rocm.repo <<EOF
[ROCm-6.4.1]
name=ROCm6.4.1
baseurl=https://repo.radeon.com/rocm/el9/6.4.1/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
sudo dnf clean all
```

OL 8

```
sudo tee --append /etc/yum.repos.d/rocm.repo <<EOF
[ROCm-6.4.1]
name=ROCm6.4.1
baseurl=https://repo.radeon.com/rocm/el8/6.4.1/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
sudo dnf clean all
```

2.2.2.4.2 Installing

2.2.2.4.2.1 Install kernel driver

For information about the AMDGPU driver installation, see the Oracle Linux native installation in the AMD Instinct Data Center GPU Documentation.

For information about driver compatibility, see User and kernel-space support matrix.

2.2.2.4.2.2 Install ROCm

```
sudo dnf install rocm
```

Complete the Post-installation instructions.

2.2.2.4.3 Upgrading

To upgrade an existing ROCm installation to a newer version, follow the steps in Registering ROCm repositories and Installing.

> **Note**
>
> Upgrading the kernel driver may also upgrade the GPU firmware, which requires a system reboot to take effect.

2.2.2.4.4 Uninstalling

2.2.2.4.4.1 Uninstall specific meta packages

```
sudo dnf remove rocm
```

2.2.2.4.4.2 Uninstall ROCm packages

```
sudo dnf remove rocm-core amdgpu-core
```

2.2.2.4.4.3 Remove ROCm repositories

```
# Remove the repositories
sudo rm /etc/yum.repos.d/rocm.repo*

# Clear the cache and clean the system
sudo rm -rf /var/cache/dnf
sudo dnf clean all

# Restart the system
sudo reboot
```

2.2.2.5 SUSE Linux Enterprise native installation

> **Caution**
>
> Ensure that the Installation prerequisites are met before installing.

2.2.2.5.1 Registering ROCm repositories

```
sudo tee --append /etc/zypp/repos.d/rocm.repo <<EOF
[ROCm-6.4.1]
```

(continues on next page)

(continued from previous page)

```
name=ROCm6.4.1
baseurl=https://repo.radeon.com/rocm/zyp/6.4.1/main
enabled=1
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF

sudo zypper refresh
```

2.2.2.5.2 Installing

2.2.2.5.2.1 Install kernel driver

For information about the AMDGPU driver installation, see the SUSE Linux Enterprise Server native installation in the AMD Instinct Data Center GPU Documentation.

For information about driver compatibility, see User and kernel-space support matrix.

2.2.2.5.2.2 Install ROCm

```
sudo zypper --gpg-auto-import-keys install rocm
```

Complete the Post-installation instructions.

2.2.2.5.3 Upgrading

To upgrade an existing ROCm installation to a newer version, follow the steps in Registering ROCm repositories and Installing.

> **Note**
>
> Upgrading the kernel driver may also upgrade the GPU firmware, which requires a system reboot to take effect.

2.2.2.5.4 Uninstalling

2.2.2.5.4.1 Uninstall specific meta packages

```
sudo zypper remove rocm
```

2.2.2.5.4.2 Uninstall ROCm packages

```
sudo zypper remove rocm-core amdgpu-core
```

2.2.2.5.4.3 Remove ROCm repositories

```
# Remove the repositories
sudo zypper removerepo "ROCm-6.4.1"
```

(continues on next page)

(continued from previous page)

```
# Clear cache and clean system
sudo zypper clean --all
sudo zypper refresh

# Restart the system
sudo reboot
```

2.2.2.6 Azure Linux native installation

> **Caution**
>
> Ensure that the Installation prerequisites are met before installing.

2.2.2.6.1 Registering ROCm repositories

AZL 3.0

```
sudo tee --append /etc/yum.repos.d/rocm.repo <<EOF
[ROCm-6.4.1]
name=ROCm6.4.1
baseurl=https://repo.radeon.com/rocm/azurelinux3/6.4.1/main/
enabled=1
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
sudo tdnf clean all
```

2.2.2.6.2 Installing

2.2.2.6.2.1 Install kernel driver

For information about the AMDGPU driver installation, see AMDGPU driver installation in the ROCm documentation.

For information about driver compatibility, see User and kernel-space support matrix.

2.2.2.6.2.2 Install ROCm

```
sudo tdnf install rocm
```

Complete the Post-installation instructions.

2.2.2.6.3 Upgrading

To upgrade an existing ROCm installation to a newer version, follow the steps in Registering ROCm repositories and Installing.

> **Note**

Upgrading the kernel driver may also upgrade the GPU firmware, which requires a system reboot to take effect.

2.2.2.6.4 Uninstalling

2.2.2.6.4.1 Uninstall specific meta packages

```
sudo tdnf remove rocm
```

2.2.2.6.4.2 Uninstall ROCm packages

```
sudo tdnf remove rocm-core
```

2.2.2.6.4.3 Remove ROCm repositories

```
# Remove the repositories
sudo rm /etc/yum.repos.d/rocm.repo*

# Clear the cache and clean the system
sudo rm -rf /var/cache/tdnf
sudo tdnf clean all

# Restart the system
sudo reboot
```

2.2.3 Installation via AMDGPU installer

Select the install instructions for your operating system

Install

- Ubuntu
- Debian
- Red Hat Enterprise Linux
- Oracle Linux
- SUSE Linux Enterprise Server

Uninstall

- Ubuntu
- Debian
- Red Hat Enterprise Linux
- Oracle Linux
- SUSE Linux Enterprise Server

See also: System requirements (Linux). If you encounter install issues, you can refer to the troubleshooting page.

2.2.3.1 Ubuntu AMDGPU installer installation

amdgpu-install is a tool that helps you install and update AMDGPU, ROCm, and ROCm components.

> **Note**
>
> ROCm doesn't support integrated graphics. If your system has an AMD IGP installed, disable it in the BIOS prior to using ROCm. If the driver can enumerate the IGP, the ROCm runtime might crash the system, even if told to omit it via HIP_VISIBLE_DEVICES.

2.2.3.1.1 Installation

> **Caution**
>
> Ensure that the Installation prerequisites are met before installing.

Ubuntu 24.04

```
sudo apt update
wget https://repo.radeon.com/amdgpu-install/6.4.1/ubuntu/noble/amdgpu-install_6.4.60401-1_all.deb
sudo apt install ./amdgpu-install_6.4.60401-1_all.deb
sudo apt update
```

Ubuntu 22.04

```
sudo apt update
wget https://repo.radeon.com/amdgpu-install/6.4.1/ubuntu/jammy/amdgpu-install_6.4.60401-1_all.deb
sudo apt install ./amdgpu-install_6.4.60401-1_all.deb
sudo apt update
```

2.2.3.1.2 Use cases

Instead of installing individual applications or libraries, the installer script groups packages into specific use cases that match typical workflows and runtimes.

To display a list of available use cases, run:

```
sudo amdgpu-install --list-usecase
```

The available use-cases are printed in a format similar to:

```
If --usecase option is not present, the default selection is "graphics,opencl,hip"

Available use cases:
dkms            (to only install the kernel mode driver)
  - Kernel mode driver (included in all usecases)
graphics        (for users of graphics applications)
  - Open source Mesa 3D graphics and multimedia libraries
multimedia      (for users of open source multimedia)
  - Open source Mesa 3D multimedia libraries
multimediasdk   (for developers of open source multimedia)
```

(continues on next page)

(continued from previous page)

```
  - Open source Mesa 3D multimedia libraries
  - Development headers for multimedia libraries
workstation     (for users of legacy WS applications)
  - Open source multimedia libraries
  - Closed source (legacy) OpenGL
rocm            (for users and developers requiring full ROCm stack)
  - OpenCL (ROCr/KMD based) runtime
  - HIP runtimes
  - Machine learning framework
  - All ROCm libraries and applications
rocmdev         (for developers requiring ROCm runtime and
                profiling/debugging tools)
  - HIP runtimes
  - OpenCL runtime
  - Profiler, Tracer and Debugger tools
rocmdevtools    (for developers requiring ROCm profiling/debugging tools)
  - Profiler, Tracer and Debugger tools
amf             (for users of AMF based multimedia)
  - AMF closed source multimedia library
lrt             (for users of applications requiring ROCm runtime)
  - ROCm Compiler and device libraries
  - ROCr runtime and thunk
opencl          (for users of applications requiring OpenCL on Vega or later
                products)
  - ROCr based OpenCL
  - ROCm Language runtime
openclsdk       (for application developers requiring ROCr based OpenCL)
  - ROCr based OpenCL
  - ROCm Language runtime
  - development and SDK files for ROCr based OpenCL
hip             (for users of HIP runtime on AMD products)
  - HIP runtimes
hiplibsdk       (for application developers requiring HIP on AMD products)
  - HIP runtimes
  - ROCm math libraries
  - HIP development libraries
openmpsdk       (for users of openmp/flang on AMD products)
  - OpenMP runtime and devel packages
mllib           (for users executing machine learning workloads)
  - MIOpen hip/tensile libraries
  - Clang OpenCL
  - MIOpen kernels
mlsdk           (for developers executing machine learning workloads)
  - MIOpen development libraries
  - Clang OpenCL development libraries
  - MIOpen kernels
asan            (for users of ASAN enabled ROCm packages)
  - ASAN enabled OpenCL (ROCr/KMD based) runtime
  - ASAN enabled HIP runtimes
  - ASAN enabled Machine learning framework
  - ASAN enabled ROCm libraries
```

2.2.3.1.3 Upgrading ROCm

The upgrade procedure with the installer script is the same as installing it for first-time use.

2.2.3.1.4 Installing ROCm packages

To install use cases specific to your requirements, use the installer (amdgpu-install) as follows:

- To install a single use case, add it with the --usecase option:

```
sudo amdgpu-install --usecase=rocm --no-dkms
```

- For multiple use cases, separate them with commas:

```
sudo amdgpu-install --usecase=hiplibsdk,rocm --no-dkms
```

- For graphical workloads using the open-source driver, add graphics. For example:

```
sudo amdgpu-install --usecase=graphics,rocm --no-dkms
```

- For graphical workloads using the proprietary driver, add workstation. For example:

```
sudo amdgpu-install --usecase=workstation,rocm --no-dkms
```

- To install LLVM AddressSanitizer (ASAN) instrumented binaries (for packages that support it), add asan. For example:

```
sudo amdgpu-install --usecase=rocm,asan --no-dkms
```

- To list all possible use cases, use the --list-usecase option:

```
sudo amdgpu-install --list-usecase
```

- The --help option displays all available options for the amdgpu-install script:

```
sudo amdgpu-install --help
```

> **Note**
>
> For information about the AMDGPU driver installation, see the Install AMDGPU driver in the AMD Instinct Data Center GPU Documentation.

2.2.3.1.5 Uninstalling

2.2.3.1.5.1 Uninstalling amdgpu-install

```
sudo apt purge amdgpu-install
sudo apt autoremove
```

2.2.3.1.5.2 Uninstall specific meta packages

```
sudo apt autoremove rocm
```

2.2.3.1.5.3 Uninstall ROCm packages

```
sudo apt autoremove rocm-core
```

2.2.3.1.5.4 Cache cleanup

```
# Clear the cache and clean the system
sudo rm -rf /var/cache/apt/*
sudo apt clean all
sudo apt update

# Restart the system
sudo reboot
```

2.2.3.1.6 Additional options

- Unattended installation.

  Adding -y as a parameter to amdgpu-install skips user prompts (for automation). For example:

  ```
  amdgpu-install -y --usecase=rocm
  ```

2.2.3.2 Debian AMDGPU installer installation

amdgpu-install is a tool that helps you install and update AMDGPU, ROCm, and ROCm components.

> **Note**
>
> ROCm doesn't support integrated graphics. If your system has an AMD IGP installed, disable it in the BIOS prior to using ROCm. If the driver can enumerate the IGP, the ROCm runtime might crash the system, even if told to omit it via HIP_VISIBLE_DEVICES.

2.2.3.2.1 Installation

> **Caution**
>
> Ensure that the Installation prerequisites are met before installing.

Debian 12

```
sudo apt update
wget https://repo.radeon.com/amdgpu-install/6.4.1/ubuntu/jammy/amdgpu-install_6.4.60401-1_all.deb
sudo apt install ./amdgpu-install_6.4.60401-1_all.deb
sudo apt update
```

2.2.3.2.2 Use cases

Instead of installing individual applications or libraries, the installer script groups packages into specific use cases that match typical workflows and runtimes.

To display a list of available use cases, run:

```
sudo amdgpu-install --list-usecase
```

The available use-cases are printed in a format similar to:

```
If --usecase option is not present, the default selection is "graphics,opencl,hip"

Available use cases:
dkms            (to only install the kernel mode driver)
  - Kernel mode driver (included in all usecases)
graphics        (for users of graphics applications)
  - Open source Mesa 3D graphics and multimedia libraries
multimedia      (for users of open source multimedia)
  - Open source Mesa 3D multimedia libraries
multimediasdk   (for developers of open source multimedia)
  - Open source Mesa 3D multimedia libraries
  - Development headers for multimedia libraries
workstation     (for users of legacy WS applications)
  - Open source multimedia libraries
  - Closed source (legacy) OpenGL
rocm            (for users and developers requiring full ROCm stack)
  - OpenCL (ROCr/KMD based) runtime
  - HIP runtimes
  - Machine learning framework
  - All ROCm libraries and applications
rocmdev         (for developers requiring ROCm runtime and
                profiling/debugging tools)
  - HIP runtimes
  - OpenCL runtime
  - Profiler, Tracer and Debugger tools
rocmdevtools    (for developers requiring ROCm profiling/debugging tools)
  - Profiler, Tracer and Debugger tools
amf             (for users of AMF based multimedia)
  - AMF closed source multimedia library
lrt             (for users of applications requiring ROCm runtime)
  - ROCm Compiler and device libraries
  - ROCr runtime and thunk
opencl          (for users of applications requiring OpenCL on Vega or later
                products)
  - ROCr based OpenCL
  - ROCm Language runtime
openclsdk       (for application developers requiring ROCr based OpenCL)
  - ROCr based OpenCL
  - ROCm Language runtime
  - development and SDK files for ROCr based OpenCL
hip             (for users of HIP runtime on AMD products)
  - HIP runtimes
hiplibsdk       (for application developers requiring HIP on AMD products)
  - HIP runtimes
  - ROCm math libraries
  - HIP development libraries
openmpsdk       (for users of openmp/flang on AMD products)
  - OpenMP runtime and devel packages
mllib           (for users executing machine learning workloads)
```

(continues on next page)

(continued from previous page)

```
  - MIOpen hip/tensile libraries
  - Clang OpenCL
  - MIOpen kernels
mlsdk           (for developers executing machine learning workloads)
  - MIOpen development libraries
  - Clang OpenCL development libraries
  - MIOpen kernels
asan            (for users of ASAN enabled ROCm packages)
  - ASAN enabled OpenCL (ROCr/KMD based) runtime
  - ASAN enabled HIP runtimes
  - ASAN enabled Machine learning framework
  - ASAN enabled ROCm libraries
```

2.2.3.2.3 Upgrading ROCm

The upgrade procedure with the installer script is the same as installing it for first-time use.

2.2.3.2.4 Installing ROCm packages

To install use cases specific to your requirements, use the installer (amdgpu-install) as follows:

- To install a single use case, add it with the --usecase option:

```
sudo amdgpu-install --usecase=rocm --no-dkms
```

- For multiple use cases, separate them with commas:

```
sudo amdgpu-install --usecase=hiplibsdk,rocm --no-dkms
```

- For graphical workloads using the open-source driver, add graphics. For example:

```
sudo amdgpu-install --usecase=graphics,rocm --no-dkms
```

- For graphical workloads using the proprietary driver, add workstation. For example:

```
sudo amdgpu-install --usecase=workstation,rocm --no-dkms
```

- To install LLVM AddressSanitizer (ASAN) instrumented binaries (for packages that support it), add asan. For example:

```
sudo amdgpu-install --usecase=rocm,asan --no-dkms
```

- To list all possible use cases, use the --list-usecase option:

```
sudo amdgpu-install --list-usecase
```

- The --help option displays all available options for the amdgpu-install script:

```
sudo amdgpu-install --help
```

Note

For information about the AMDGPU driver installation, see the Install AMDGPU driver in the AMD Instinct Data Center GPU Documentation.

2.2.3.2.5 Uninstalling

2.2.3.2.5.1 Uninstalling amdgpu-install

```
sudo apt purge amdgpu-install
sudo apt autoremove
```

2.2.3.2.5.2 Uninstall specific meta packages

```
sudo apt autoremove rocm
```

2.2.3.2.5.3 Uninstall ROCm packages

```
sudo apt autoremove rocm-core
```

2.2.3.2.5.4 Cache cleanup

```
# Clear the cache and clean the system
sudo rm -rf /var/cache/apt/*
sudo apt clean all
sudo apt update

# Restart the system
sudo reboot
```

2.2.3.2.6 Additional options

- Unattended installation.

  Adding -y as a parameter to amdgpu-install skips user prompts (for automation). For example:

  ```
  amdgpu-install -y --usecase=rocm
  ```

2.2.3.3 Red Hat Enterprise Linux AMDGPU installer installation

amdgpu-install is a tool that helps you install and update AMDGPU, ROCm, and ROCm components.

Note

ROCm doesn't support integrated graphics. If your system has an AMD IGP installed, disable it in the BIOS prior to using ROCm. If the driver can enumerate the IGP, the ROCm runtime might crash the system, even if told to omit it via HIP_VISIBLE_DEVICES.

2.2.3.3.1 Installation

> **Caution**
>
> Ensure that the Installation prerequisites are met before installing.

RHEL 9.6

```
sudo dnf install https://repo.radeon.com/amdgpu-install/6.4.1/rhel/9.6/amdgpu-install-6.4.60401-1.el9.
↪noarch.rpm
```

RHEL 9.5

```
sudo dnf install https://repo.radeon.com/amdgpu-install/6.4.1/rhel/9.5/amdgpu-install-6.4.60401-1.el9.
↪noarch.rpm
```

RHEL 9.4

```
sudo dnf install https://repo.radeon.com/amdgpu-install/6.4.1/rhel/9.4/amdgpu-install-6.4.60401-1.el9.
↪noarch.rpm
```

RHEL 8.10

```
sudo dnf install https://repo.radeon.com/amdgpu-install/6.4.1/rhel/8.10/amdgpu-install-6.4.60401-1.el8.
↪noarch.rpm
```

2.2.3.3.2 Use cases

Instead of installing individual applications or libraries, the installer script groups packages into specific use cases that match typical workflows and runtimes.

To display a list of available use cases, run:

```
sudo amdgpu-install --list-usecase
```

The available use-cases are printed in a format similar to:

```
If --usecase option is not present, the default selection is "graphics,opencl,hip"

Available use cases:
dkms            (to only install the kernel mode driver)
  - Kernel mode driver (included in all usecases)
graphics        (for users of graphics applications)
  - Open source Mesa 3D graphics and multimedia libraries
multimedia      (for users of open source multimedia)
  - Open source Mesa 3D multimedia libraries
multimediasdk   (for developers of open source multimedia)
  - Open source Mesa 3D multimedia libraries
  - Development headers for multimedia libraries
workstation     (for users of legacy WS applications)
  - Open source multimedia libraries
```

(continues on next page)

(continued from previous page)

```
  - Closed source (legacy) OpenGL
rocm            (for users and developers requiring full ROCm stack)
  - OpenCL (ROCr/KMD based) runtime
  - HIP runtimes
  - Machine learning framework
  - All ROCm libraries and applications
rocmdev         (for developers requiring ROCm runtime and
                profiling/debugging tools)
  - HIP runtimes
  - OpenCL runtime
  - Profiler, Tracer and Debugger tools
rocmdevtools    (for developers requiring ROCm profiling/debugging tools)
  - Profiler, Tracer and Debugger tools
amf             (for users of AMF based multimedia)
  - AMF closed source multimedia library
lrt             (for users of applications requiring ROCm runtime)
  - ROCm Compiler and device libraries
  - ROCr runtime and thunk
opencl          (for users of applications requiring OpenCL on Vega or later
                products)
  - ROCr based OpenCL
  - ROCm Language runtime
openclsdk       (for application developers requiring ROCr based OpenCL)
  - ROCr based OpenCL
  - ROCm Language runtime
  - development and SDK files for ROCr based OpenCL
hip             (for users of HIP runtime on AMD products)
  - HIP runtimes
hiplibsdk       (for application developers requiring HIP on AMD products)
  - HIP runtimes
  - ROCm math libraries
  - HIP development libraries
openmpsdk       (for users of openmp/flang on AMD products)
  - OpenMP runtime and devel packages
mllib           (for users executing machine learning workloads)
  - MIOpen hip/tensile libraries
  - Clang OpenCL
  - MIOpen kernels
mlsdk           (for developers executing machine learning workloads)
  - MIOpen development libraries
  - Clang OpenCL development libraries
  - MIOpen kernels
asan            (for users of ASAN enabled ROCm packages)
  - ASAN enabled OpenCL (ROCr/KMD based) runtime
  - ASAN enabled HIP runtimes
  - ASAN enabled Machine learning framework
  - ASAN enabled ROCm libraries
```

2.2.3.3.3 Upgrading ROCm

The upgrade procedure with the installer script is the same as installing it for first-time use.

2.2.3.3.4 Installing ROCm packages

To install use cases specific to your requirements, use the installer (amdgpu-install) as follows:

- To install a single use case, add it with the --usecase option:

```
sudo amdgpu-install --usecase=rocm --no-dkms
```

- For multiple use cases, separate them with commas:

```
sudo amdgpu-install --usecase=hiplibsdk,rocm --no-dkms
```

- For graphical workloads using the open-source driver, add graphics. For example:

```
sudo amdgpu-install --usecase=graphics,rocm --no-dkms
```

- For graphical workloads using the proprietary driver, add workstation. For example:

```
sudo amdgpu-install --usecase=workstation,rocm --no-dkms
```

- To install LLVM AddressSanitizer (ASAN) instrumented binaries (for packages that support it), add asan. For example:

```
sudo amdgpu-install --usecase=rocm,asan --no-dkms
```

- To list all possible use cases, use the --list-usecase option:

```
sudo amdgpu-install --list-usecase
```

- The --help option displays all available options for the amdgpu-install script:

```
sudo amdgpu-install --help
```

> **Note**
>
> For information about the AMDGPU driver installation, see the Install AMDGPU driver in the AMD Instinct Data Center GPU Documentation.

2.2.3.3.5 Uninstalling

2.2.3.3.5.1 Uninstalling amdgpu-install

```
sudo dnf remove amdgpu-install
```

2.2.3.3.5.2 Uninstall specific meta packages

```
sudo dnf remove rocm
```

2.2.3.3.5.3 Uninstall ROCm packages

```
sudo dnf remove rocm-core amdgpu-core
```

2.2.3.3.5.4 Cache cleanup

```
# Clear the cache and clean the system
sudo rm -rf /var/cache/dnf
sudo dnf clean all

# Restart the system
sudo reboot
```

2.2.3.3.6 Additional options

- Unattended installation.

  Adding -y as a parameter to amdgpu-install skips user prompts (for automation). For example:

  ```
  amdgpu-install -y --usecase=rocm
  ```

2.2.3.4 Oracle Linux AMDGPU installer installation

amdgpu-install is a tool that helps you install and update AMDGPU, ROCm, and ROCm components.

> **Note**
>
> ROCm doesn't support integrated graphics. If your system has an AMD IGP installed, disable it in the BIOS prior to using ROCm. If the driver can enumerate the IGP, the ROCm runtime might crash the system, even if told to omit it via HIP_VISIBLE_DEVICES.

2.2.3.4.1 Installation

> **Caution**
>
> Ensure that the Installation prerequisites are met before installing.

OL 9

```
sudo dnf install https://repo.radeon.com/amdgpu-install/6.4.1/el/9.5/amdgpu-install-6.4.60401-1.el9.
↪noarch.rpm
```

OL 8

```
sudo dnf install https://repo.radeon.com/amdgpu-install/6.4.1/el/8.10/amdgpu-install-6.4.60401-1.el8.
↪noarch.rpm
```

2.2.3.4.2 Use cases

Instead of installing individual applications or libraries, the installer script groups packages into specific use cases that match typical workflows and runtimes.

To display a list of available use cases, run:

```
sudo amdgpu-install --list-usecase
```

The available use-cases are printed in a format similar to:

```
If --usecase option is not present, the default selection is "graphics,opencl,hip"

Available use cases:
dkms            (to only install the kernel mode driver)
  - Kernel mode driver (included in all usecases)
graphics        (for users of graphics applications)
  - Open source Mesa 3D graphics and multimedia libraries
multimedia      (for users of open source multimedia)
  - Open source Mesa 3D multimedia libraries
multimediasdk   (for developers of open source multimedia)
  - Open source Mesa 3D multimedia libraries
  - Development headers for multimedia libraries
workstation     (for users of legacy WS applications)
  - Open source multimedia libraries
  - Closed source (legacy) OpenGL
rocm            (for users and developers requiring full ROCm stack)
  - OpenCL (ROCr/KMD based) runtime
  - HIP runtimes
  - Machine learning framework
  - All ROCm libraries and applications
rocmdev         (for developers requiring ROCm runtime and
                profiling/debugging tools)
  - HIP runtimes
  - OpenCL runtime
  - Profiler, Tracer and Debugger tools
rocmdevtools    (for developers requiring ROCm profiling/debugging tools)
  - Profiler, Tracer and Debugger tools
amf             (for users of AMF based multimedia)
  - AMF closed source multimedia library
lrt             (for users of applications requiring ROCm runtime)
  - ROCm Compiler and device libraries
  - ROCr runtime and thunk
opencl          (for users of applications requiring OpenCL on Vega or later
                products)
  - ROCr based OpenCL
  - ROCm Language runtime
openclsdk       (for application developers requiring ROCr based OpenCL)
  - ROCr based OpenCL
  - ROCm Language runtime
  - development and SDK files for ROCr based OpenCL
hip             (for users of HIP runtime on AMD products)
  - HIP runtimes
hiplibsdk       (for application developers requiring HIP on AMD products)
```

(continues on next page)

(continued from previous page)

```
  - HIP runtimes
  - ROCm math libraries
  - HIP development libraries
openmpsdk       (for users of openmp/flang on AMD products)
  - OpenMP runtime and devel packages
mllib           (for users executing machine learning workloads)
  - MIOpen hip/tensile libraries
  - Clang OpenCL
  - MIOpen kernels
mlsdk           (for developers executing machine learning workloads)
  - MIOpen development libraries
  - Clang OpenCL development libraries
  - MIOpen kernels
asan            (for users of ASAN enabled ROCm packages)
  - ASAN enabled OpenCL (ROCr/KMD based) runtime
  - ASAN enabled HIP runtimes
  - ASAN enabled Machine learning framework
  - ASAN enabled ROCm libraries
```

2.2.3.4.3 Upgrading ROCm

The upgrade procedure with the installer script is the same as installing it for first-time use.

2.2.3.4.4 Installing ROCm packages

To install use cases specific to your requirements, use the installer (amdgpu-install) as follows:

- To install a single use case, add it with the --usecase option:

```
sudo amdgpu-install --usecase=rocm --no-dkms
```

- For multiple use cases, separate them with commas:

```
sudo amdgpu-install --usecase=hiplibsdk,rocm --no-dkms
```

- For graphical workloads using the open-source driver, add graphics. For example:

```
sudo amdgpu-install --usecase=graphics,rocm --no-dkms
```

- For graphical workloads using the proprietary driver, add workstation. For example:

```
sudo amdgpu-install --usecase=workstation,rocm --no-dkms
```

- To install LLVM AddressSanitizer (ASAN) instrumented binaries (for packages that support it), add asan. For example:

```
sudo amdgpu-install --usecase=rocm,asan --no-dkms
```

- To list all possible use cases, use the --list-usecase option:

```
sudo amdgpu-install --list-usecase
```

- The --help option displays all available options for the amdgpu-install script:

```
sudo amdgpu-install --help
```

> **Note**
>
> For information about the AMDGPU driver installation, see the Install AMDGPU driver in the AMD Instinct Data Center GPU Documentation.

2.2.3.4.5 Uninstalling

2.2.3.4.5.1 Uninstalling amdgpu-install

```
sudo dnf remove amdgpu-install
```

2.2.3.4.5.2 Uninstall specific meta packages

```
sudo dnf remove rocm
```

2.2.3.4.5.3 Uninstall ROCm packages

```
sudo dnf remove rocm-core amdgpu-core
```

2.2.3.4.6 Cache cleanup

```
# Clear the cache and clean the system
sudo rm -rf /var/cache/dnf
sudo dnf clean all

# Restart the system
sudo reboot
```

2.2.3.4.7 Additional options

- Unattended installation.

  Adding -y as a parameter to amdgpu-install skips user prompts (for automation). For example:

  ```
  amdgpu-install -y --usecase=rocm
  ```

2.2.3.5 SUSE Enterprise Linux AMDGPU installer installation

amdgpu-install is a tool that helps you install and update AMDGPU, ROCm, and ROCm components.

> **Note**
>
> ROCm doesn't support integrated graphics. If your system has an AMD IGP installed, disable it in the BIOS prior to using ROCm. If the driver can enumerate the IGP, the ROCm runtime might crash the system, even if told to omit it via HIP_VISIBLE_DEVICES.

2.2.3.5.1 Installation

> **Caution**
>
> Ensure that the Installation prerequisites are met before installing.

SLES 15.6

```
sudo zypper --no-gpg-checks install https://repo.radeon.com/amdgpu-install/6.4.1/sle/15.6/amdgpu-
↪install-6.4.60401-1.noarch.rpm
```

2.2.3.5.2 Use cases

Instead of installing individual applications or libraries, the installer script groups packages into specific use cases that match typical workflows and runtimes.

To display a list of available use cases, run:

```
sudo amdgpu-install --list-usecase
```

The available use-cases are printed in a format similar to:

```
If --usecase option is not present, the default selection is "graphics,opencl,hip"

Available use cases:
dkms             (to only install the kernel mode driver)
  - Kernel mode driver (included in all usecases)
graphics         (for users of graphics applications)
  - Open source Mesa 3D graphics and multimedia libraries
multimedia       (for users of open source multimedia)
  - Open source Mesa 3D multimedia libraries
multimediasdk    (for developers of open source multimedia)
  - Open source Mesa 3D multimedia libraries
  - Development headers for multimedia libraries
workstation      (for users of legacy WS applications)
  - Open source multimedia libraries
  - Closed source (legacy) OpenGL
rocm             (for users and developers requiring full ROCm stack)
  - OpenCL (ROCr/KMD based) runtime
  - HIP runtimes
  - Machine learning framework
  - All ROCm libraries and applications
rocmdev          (for developers requiring ROCm runtime and
                  profiling/debugging tools)
  - HIP runtimes
  - OpenCL runtime
  - Profiler, Tracer and Debugger tools
rocmdevtools     (for developers requiring ROCm profiling/debugging tools)
  - Profiler, Tracer and Debugger tools
amf              (for users of AMF based multimedia)
  - AMF closed source multimedia library
lrt              (for users of applications requiring ROCm runtime)
```

(continues on next page)

(continued from previous page)

```
  - ROCm Compiler and device libraries
  - ROCr runtime and thunk
opencl          (for users of applications requiring OpenCL on Vega or later
                products)
  - ROCr based OpenCL
  - ROCm Language runtime
openclsdk       (for application developers requiring ROCr based OpenCL)
  - ROCr based OpenCL
  - ROCm Language runtime
  - development and SDK files for ROCr based OpenCL
hip             (for users of HIP runtime on AMD products)
  - HIP runtimes
hiplibsdk       (for application developers requiring HIP on AMD products)
  - HIP runtimes
  - ROCm math libraries
  - HIP development libraries
openmpsdk       (for users of openmp/flang on AMD products)
  - OpenMP runtime and devel packages
mllib           (for users executing machine learning workloads)
  - MIOpen hip/tensile libraries
  - Clang OpenCL
  - MIOpen kernels
mlsdk           (for developers executing machine learning workloads)
  - MIOpen development libraries
  - Clang OpenCL development libraries
  - MIOpen kernels
asan            (for users of ASAN enabled ROCm packages)
  - ASAN enabled OpenCL (ROCr/KMD based) runtime
  - ASAN enabled HIP runtimes
  - ASAN enabled Machine learning framework
  - ASAN enabled ROCm libraries
```

2.2.3.5.3 Upgrading ROCm

The upgrade procedure with the installer script is the same as installing it for first-time use.

2.2.3.5.4 Installing ROCm packages

To install use cases specific to your requirements, use the installer (amdgpu-install) as follows:

- To install a single use case, add it with the --usecase option:

```
sudo amdgpu-install --usecase=rocm --no-dkms
```

- For multiple use cases, separate them with commas:

```
sudo amdgpu-install --usecase=hiplibsdk,rocm --no-dkms
```

- For graphical workloads using the open-source driver, add graphics. For example:

```
sudo amdgpu-install --usecase=graphics,rocm --no-dkms
```

- For graphical workloads using the proprietary driver, add workstation. For example:

```
sudo amdgpu-install --usecase=workstation,rocm --no-dkms
```

- To install LLVM AddressSanitizer (ASAN) instrumented binaries (for packages that support it), add asan. For example:

```
sudo amdgpu-install --usecase=rocm,asan --no-dkms
```

- To list all possible use cases, use the --list-usecase option:

```
sudo amdgpu-install --list-usecase
```

- The --help option displays all available options for the amdgpu-install script:

```
sudo amdgpu-install --help
```

> **Note**
>
> For information about the AMDGPU driver installation, see the Install AMDGPU driver in the AMD Instinct Data Center GPU Documentation.

2.2.3.5.5 Uninstalling

2.2.3.5.5.1 Uninstalling amdgpu-install

```
sudo zypper remove amdgpu-install
```

2.2.3.5.5.2 Uninstall specific meta packages

```
sudo zypper remove rocm
```

2.2.3.5.5.3 Uninstall ROCm packages

```
sudo zypper remove rocm-core amdgpu-core
```

2.2.3.5.5.4 Cache cleanup

```
# Clear the cache and clean the system
sudo zypper clean --all
sudo zypper refresh

# Restart the system
sudo reboot
```

2.2.3.5.6 Additional options

- Unattended installation.

  Adding -y as a parameter to amdgpu-install skips user prompts (for automation). For example:

```
amdgpu-install -y --usecase=rocm
```

2.2.4 Installing multiple ROCm versions

A multi-version ROCm installation covers situations where you need multiple versions of ROCm on the same machine–for compatibility with different applications and hardware, testing, and other use cases.

A multi-version ROCm installation involves the following:

- Installing multiple instances of the ROCm stack on a system.
- Using versioned ROCm meta-packages. ROCm packages are versioned with both a ROCm release version and package-specific semantic versioning. Extending a package name and its dependencies with the release version adds the ability to support multiple versions of packages simultaneously.

A single-version ROCm installation involves the following.

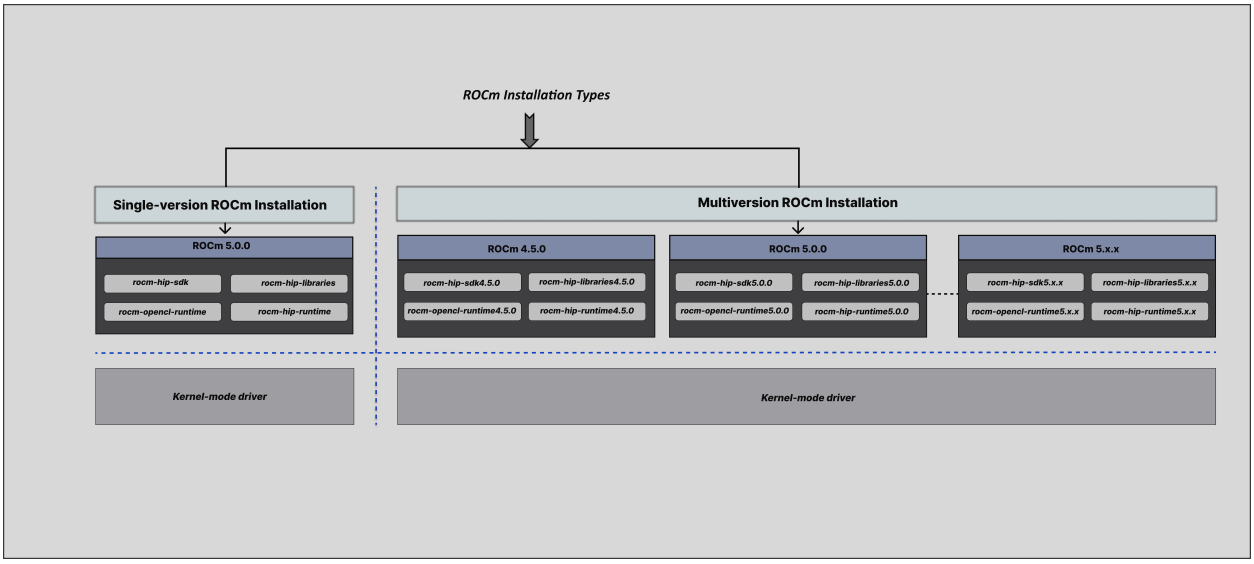
- Installing a single instance of the ROCm release on a system.
- Using non-versioned ROCm meta-packages.

See Quick start installation guide or ROCm on Linux detailed installation overview for a standard single-version installation.

Caution

You cannot install single-version and multi-version ROCm packages together on the same machine. The conflicting package versions might result in unpredictable behavior.

The following illustrations shows the difference between single-version and multi-version ROCm installations.



Select the install and uninstall instructions for your operating system

Install

- Ubuntu
- Debian
- Red Hat Enterprise Linux

- Oracle Linux
- SUSE Linux Enterprise Server
- Azure Linux

Uninstall

- Ubuntu
- Debian
- Red Hat Enterprise Linux
- Oracle Linux
- SUSE Linux Enterprise Server
- Azure Linux

See also: System requirements (Linux). If you encounter install issues, you can refer to the troubleshooting page.

2.2.4.1 Ubuntu multi-version installation

Caution

Ensure that the Installation prerequisites are met before installing.

2.2.4.1.1 Registering ROCm repositories

2.2.4.1.1.1 Package signing key

Download and convert the package signing key.

```
# Make the directory if it doesn't exist yet.
# This location is recommended by the distribution maintainers.
sudo mkdir --parents --mode=0755 /etc/apt/keyrings

# Download the key, convert the signing-key to a full
# keyring required by apt and store in the keyring directory
wget https://repo.radeon.com/rocm/rocm.gpg.key -O - | \
gpg --dearmor | sudo tee /etc/apt/keyrings/rocm.gpg > /dev/null
```

Note

The GPG key may change; ensure it is updated when installing a new release. If the key signature verification fails while updating, re-add the key from the ROCm to the apt repository as mentioned above.

2.2.4.1.1.2 Register packages

Ubuntu 24.04

```
echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/amdgpu/6.4.1/
↪ubuntu noble main" \
```

(continues on next page)

(continued from previous page)

```
    | sudo tee /etc/apt/sources.list.d/amdgpu.list

# Note: There is NO trailing .0 in the patch version when registering repositories
for ver in 6.4.1 6.4; do
echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/rocm/apt/$ver␣
↪noble main" \
    | sudo tee --append /etc/apt/sources.list.d/rocm.list
done
echo -e 'Package: *\nPin: release o=repo.radeon.com\nPin-Priority: 600' \
    | sudo tee /etc/apt/preferences.d/rocm-pin-600
sudo apt update
```

Ubuntu 22.04

```
echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/amdgpu/6.4.1/
↪ubuntu jammy main" \
    | sudo tee /etc/apt/sources.list.d/amdgpu.list

# Note: There is NO trailing .0 in the patch version when registering repositories
for ver in 6.4.1 6.4; do
echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/rocm/apt/$ver␣
↪jammy main" \
    | sudo tee --append /etc/apt/sources.list.d/rocm.list
done
echo -e 'Package: *\nPin: release o=repo.radeon.com\nPin-Priority: 600' \
    | sudo tee /etc/apt/preferences.d/rocm-pin-600
sudo apt update
```

2.2.4.1.2 Installing

2.2.4.1.2.1 Install kernel driver

For information about the AMDGPU driver installation, see the Ubuntu native installation in the AMD Instinct Data Center GPU Documentation.

For information about driver compatibility, see User and kernel-space support matrix.

2.2.4.1.2.2 Install ROCm

Before proceeding with a multi-version ROCm installation, you must remove ROCm packages that were previously installed from a single-version installation to avoid conflicts.

```
# Note: There IS a trailing .0 in the patch version for packages
for ver in 6.4.1 6.4.0; do
    sudo apt install rocm$ver
done
```

> **Note**
>
> For versions earlier than ROCm 6.0.0, use rocm-hip-sdk instead of rocm (for example, rocm-hip-sdk5.7.1).

Complete the Post-installation instructions.

> **Tip**
>
> For a single-version installation of the latest ROCm version on Ubuntu, use the steps in Registering ROCm repositories and Installing.

2.2.4.1.3 Uninstalling

2.2.4.1.3.1 Uninstall specific meta packages

```
# Note: There IS a trailing .0 in the patch version for packages
for ver in 6.4.1 6.4.0; do
    sudo apt autoremove rocm$ver
done
```

2.2.4.1.3.2 Uninstall ROCm packages

```
# Note: There IS a trailing .0 in the patch version for packages
for ver in 6.4.1 6.4.0; do
    sudo apt autoremove rocm-core$ver
done
```

2.2.4.1.3.3 Remove ROCm repositories

```
# Remove ROCm repositories
sudo rm /etc/apt/sources.list.d/rocm.list
sudo rm /etc/apt/sources.list.d/amdgpu.list

# Clear the cache and clean the system
sudo rm -rf /var/cache/apt/*
sudo apt clean all
sudo apt update

# Restart the system
sudo reboot
```

2.2.4.2 Debian multi-version installation

> **Caution**
>
> Ensure that the Installation prerequisites are met before installing.

2.2.4.2.1 Registering ROCm repositories

2.2.4.2.1.1 Package signing key

Download and convert the package signing key.

```
# Make the directory if it doesn't exist yet.
# This location is recommended by the distribution maintainers.
sudo mkdir --parents --mode=0755 /etc/apt/keyrings

# Download the key, convert the signing-key to a full
# keyring required by apt and store in the keyring directory
wget https://repo.radeon.com/rocm/rocm.gpg.key -O - | \
gpg --dearmor | sudo tee /etc/apt/keyrings/rocm.gpg > /dev/null
```

> **Note**
>
> The GPG key may change; ensure it is updated when installing a new release. If the key signature verification fails while updating, re-add the key from the ROCm to the apt repository as mentioned above.

2.2.4.2.1.2 Register packages

Debian 12

```
echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/amdgpu/6.4.1/
↪ubuntu jammy main" \
    | sudo tee /etc/apt/sources.list.d/amdgpu.list

# Note: There is NO trailing .0 in the patch version for repositories
for ver in 6.4.1 6.4; do
echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/rocm/apt/$ver␣
↪jammy main" \
    | sudo tee --append /etc/apt/sources.list.d/rocm.list
done
echo -e 'Package: *\nPin: release o=repo.radeon.com\nPin-Priority: 600' \
    | sudo tee /etc/apt/preferences.d/rocm-pin-600
sudo apt update
```

2.2.4.2.2 Installing

2.2.4.2.2.1 Install kernel driver

For information about the AMDGPU driver installation, see the Debian native installation in the AMD Instinct Data Center GPU Documentation.

For information about driver compatibility, see User and kernel-space support matrix.

2.2.4.2.2.2 Install ROCm

Before proceeding with a multi-version ROCm installation, you must remove ROCm packages that were previously installed from a single-version installation to avoid conflicts.

```
# Note: There IS a trailing .0 in the patch version for packages
for ver in 6.4.1 6.4.0; do
    sudo apt install rocm$ver
done
```

> **Note**
>
> For versions earlier than ROCm 6.0.0, use rocm-hip-sdk instead of rocm (for example, rocm-hip-sdk5.7.1).

Complete the Post-installation instructions.

> **Tip**
>
> For a single-version installation of the latest ROCm version on Debian, use the steps in Registering ROCm repositories and Installing.

2.2.4.2.3 Uninstalling

2.2.4.2.3.1 Uninstall specific meta packages

```
# Note: There IS a trailing .0 in the patch version for packages
for ver in 6.4.1 6.4.0; do
    sudo apt autoremove rocm$ver
done
```

2.2.4.2.3.2 Uninstall ROCm packages

```
# Note: There IS a trailing .0 in the patch version for packages
for ver in 6.4.1 6.4.0; do
    sudo apt autoremove rocm-core$ver
done
```

2.2.4.2.3.3 Remove ROCm repositories

```
# Remove ROCm and AMDGPU repositories
sudo rm /etc/apt/sources.list.d/rocm.list
sudo rm /etc/apt/sources.list.d/amdgpu.list

# Clear the cache and clean the system
sudo rm -rf /var/cache/apt/*
sudo apt clean all
sudo apt update

# Restart the system
sudo reboot
```

2.2.4.3 Red Hat Enterprise Linux multi-version installation

> **Caution**
>
> Ensure that the Installation prerequisites are met before installing.

2.2.4.3.1 Registering ROCm repositories

RHEL 9

```
# Note: There is NO trailing .0 in the patch version for repositories
for ver in 6.4.1 6.4; do
sudo tee --append /etc/yum.repos.d/rocm.repo <<EOF
[ROCm-$ver]
name=ROCm$ver
baseurl=https://repo.radeon.com/rocm/el9/$ver/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
done
sudo dnf clean all
```

RHEL 8

```
# Note: There is NO trailing .0 in the patch version for repositories
for ver in 6.4.1 6.4; do
sudo tee --append /etc/yum.repos.d/rocm.repo <<EOF
[ROCm-$ver]
name=ROCm$ver
baseurl=https://repo.radeon.com/rocm/el8/$ver/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
done
sudo dnf clean all
```

2.2.4.3.2 Installing

2.2.4.3.2.1 Install kernel driver

For information about the AMDGPU driver installation, see the Red Hat Enterprise Linux native installation in the AMD Instinct Data Center GPU Documentation.

For information about driver compatibility, see User and kernel-space support matrix.

2.2.4.3.2.2 Install ROCm

Before proceeding with a multi-version ROCm installation, you must remove ROCm packages that were previously installed from a single-version installation to avoid conflicts.

```
# Note: There IS a trailing .0 in the patch version for packages
for ver in 6.4.1 6.4.0; do
    sudo dnf install rocm$ver
done
```

Note

For versions earlier than ROCm 6.0.0, use rocm-hip-sdk instead of rocm (for example, rocm-hip-sdk5.7.1).

Complete the Post-installation instructions.

Tip

For a single-version installation of the latest ROCm version on RHEL, use the steps in Registering ROCm repositories and Installing.

2.2.4.3.3 Uninstalling

2.2.4.3.3.1 Uninstall specific meta packages

```
# Note: There IS a trailing .0 in the patch version for packages
for ver in 6.4.1 6.4.0; do
  sudo dnf remove rocm$ver
done
```

2.2.4.3.3.2 Uninstall ROCm packages

```
# Note: There IS a trailing .0 in the patch version for packages
for ver in 6.4.1 6.4.0; do
  sudo dnf remove rocm-core$ver amdgpu-core$ver
done
```

2.2.4.3.3.3 Remove ROCm repositories

```
# Remove ROCm repositories
sudo rm /etc/yum.repos.d/rocm.repo*

# Clear the cache and clean the system
sudo rm -rf /var/cache/dnf
sudo dnf clean all

# Restart the system
sudo reboot
```

2.2.4.4 Oracle Linux multi-version installation

Caution

Ensure that the Installation prerequisites are met before installing.

2.2.4.4.1 Register ROCm repositories

OL 9

```
# Note: There is NO trailing .0 in the patch version for repositories
for ver in 6.4.1 6.4; do
sudo tee --append /etc/yum.repos.d/rocm.repo <<EOF
[ROCm-$ver]
name=ROCm$ver
baseurl=https://repo.radeon.com/rocm/el9/$ver/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
done
sudo dnf clean all
```

OL 8

```
# Note: There is NO trailing .0 in the patch version for repositories
for ver in 6.4.1 6.4; do
sudo tee --append /etc/yum.repos.d/rocm.repo <<EOF
[ROCm-$ver]
name=ROCm$ver
baseurl=https://repo.radeon.com/rocm/el8/$ver/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
done
sudo dnf clean all
```

2.2.4.4.2 Installing

2.2.4.4.2.1 Install kernel driver

For information about the AMDGPU driver installation, see the Oracle Linux native installation in the AMD Instinct Data Center GPU Documentation.

For information about driver compatibility, see User and kernel-space support matrix.

2.2.4.4.2.2 Install ROCm

Before proceeding with a multi-version ROCm installation, you must remove ROCm packages that were previously installed from a single-version installation to avoid conflicts.

```
# Note: There IS a trailing .0 in the patch version for packages
for ver in 6.4.1 6.4.0; do
    sudo dnf install rocm$ver
done
```

> **Note**
>
> For versions earlier than ROCm 6.0.0, use rocm-hip-sdk instead of rocm (for example, rocm-hip-sdk5.7.1).

Complete the Post-installation instructions.

> **Tip**
>
> For a single-version installation of the latest ROCm version on OL, use the steps in Register ROCm repositories and Installing.

2.2.4.4.3 Uninstalling

2.2.4.4.3.1 Uninstall specific meta packages

```
# Note: There IS a trailing .0 in the patch version for packages
for ver in 6.4.1 6.4.0; do
  sudo dnf remove rocm$ver
done
```

2.2.4.4.3.2 Uninstall ROCm packages

```
# Note: There IS a trailing .0 in the patch version for packages
for ver in 6.4.1 6.4.0; do
  sudo dnf remove rocm-core$ver amdgpu-core$ver
done
```

2.2.4.4.3.3 Remove ROCm repositories

```
# Remove ROCm repositories
sudo rm /etc/yum.repos.d/rocm.repo*

# Clear the cache and clean the system
sudo rm -rf /var/cache/dnf
sudo dnf clean all

# Restart the system
sudo reboot
```

2.2.4.5 SUSE Linux Enterprise multi-version installation

> **Caution**
>
> Ensure that the Installation prerequisites are met before installing.

2.2.4.5.1 Registering ROCm repositories

SLES 15.6

```
# Note: There is NO trailing .0 in the patch version for repositories
for ver in 6.4.1 6.4; do
sudo tee --append /etc/zypp/repos.d/rocm.repo <<EOF
[ROCm-$ver]
```

(continues on next page)

(continued from previous page)

```
name=ROCm$ver
baseurl=https://repo.radeon.com/rocm/zyp/$ver/main
enabled=1
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
done
sudo zypper refresh
```

2.2.4.5.2 Installing

2.2.4.5.2.1 Install kernel driver

For information about the AMDGPU driver installation, see the SUSE Linux Enterprise Server native installation in the AMD Instinct Data Center GPU Documentation.

For information about driver compatibility, see User and kernel-space support matrix.

2.2.4.5.2.2 Install ROCm

Before proceeding with a multi-version ROCm installation, you must remove ROCm packages that were previously installed from a single-version installation to avoid conflicts.

```
# Note: There IS a trailing .0 in the patch version for packages
for ver in 6.4.1 6.4.0; do
      sudo zypper --gpg-auto-import-keys install rocm$ver
done
```

> **Note**
>
> For versions earlier than ROCm 6.0.0, use rocm-hip-sdk instead of rocm (for example, rocm-hip-sdk5.7.1).

Complete the Post-installation instructions.

> **Tip**
>
> For a single-version installation of the latest ROCm version on SLES, use the steps in Registering ROCm repositories and Installing.

2.2.4.5.3 Uninstalling

2.2.4.5.3.1 Uninstall specific meta packages

```
# Note: There IS a trailing .0 in the patch version for packages
for ver in 6.4.1 6.4.0; do
   sudo zypper remove rocm$ver
done
```

2.2.4.5.3.2 Uninstall ROCm packages

```
# Note: There IS a trailing .0 in the patch version for packages
for ver in 6.4.1 6.4.0; do
  sudo zypper remove rocm-core$ver amdgpu-core$ver
done
```

2.2.4.5.3.3 Remove ROCm repositories

```
# Remove the repositories
# Note: There is NO trailing .0 in the patch version for repositories
for ver in 6.4.1 6.4; do
  sudo zypper removerepo "ROCm-$ver"
done

# Clear cache and clean system
sudo zypper clean --all
sudo zypper refresh

# Restart the system
sudo reboot
```

2.2.4.6 Azure Linux multi-version installation

> **Caution**
>
> Ensure that the Installation prerequisites are met before installing.

2.2.4.6.1 Registering ROCm repositories

AZL

```
# Note: There is NO trailing .0 in the patch version for repositories
for ver in 6.4.1 6.4; do
sudo tee --append /etc/yum.repos.d/rocm.repo <<EOF
[ROCm-$ver]
name=ROCm$ver
baseurl=https://repo.radeon.com/rocm/azurelinux3/$ver/main/
enabled=1
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
sudo tdnf clean all
```

2.2.4.6.2 Installing

2.2.4.6.2.1 Install kernel driver

For information about the AMDGPU driver installation, see AMDGPU driver installation in the ROCm documentation.

For information about driver compatibility, see User and kernel-space support matrix.

2.2.4.6.2.2 Install ROCm

Before proceeding with a multi-version ROCm installation, you must remove ROCm packages that were previously installed from a single-version installation to avoid conflicts.

```
# Note: There IS a trailing .0 in the patch version for packages
for ver in 6.4.1 6.4.0; do
      sudo tdnf install rocm$ver
done
```

> **Note**
>
> For versions earlier than ROCm 6.0.0, use rocm-hip-sdk instead of rocm (for example, rocm-hip-sdk5.7.1).

Complete the Post-installation instructions.

> **Tip**
>
> For a single-version installation of the latest ROCm version on AZL, use the steps in Registering ROCm repositories and Installing.

2.2.4.6.3 Uninstalling

2.2.4.6.3.1 Uninstall specific meta packages

```
# Note: There IS a trailing .0 in the patch version for packages
for ver in 6.4.1 6.4.0; do
   sudo tdnf remove rocm$ver
done
```

2.2.4.6.3.2 Uninstall ROCm packages

```
# Note: There IS a trailing .0 in the patch version for packages
for ver in 6.4.1 6.4.0; do
   sudo tdnf remove rocm-core$ver
done
```

2.2.4.6.3.3 Remove ROCm repositories

```
# Remove ROCm repositories
sudo rm /etc/yum.repos.d/rocm.repo*

# Clear the cache and clean the system
sudo rm -rf /var/cache/tdnf
sudo tdnf clean all

# Restart the system
sudo reboot
```

2.2.5 ROCm Offline Installer Creator

The ROCm Offline Installer Creator creates an installation package for a preconfigured setup of ROCm, the AMDGPU driver, or a combination of the two on a target system without network or internet access.

On a system with internet access (known as the host), the tool creates an installer package. It uses the host system as the template for a matching offline system (known as a target). After creating the offline installer package, you can run it on a target without network access to install ROCm and the AMDGPU driver.

The ROCm Offline Installer Creator lets you customize multiple unique configurations for use when installing ROCm on a target.

The ROCm Offline Installer Creator features:

- A lightweight user interface
- An easy-to-use user interface for configuring the creation of the installer
- Support for multiple Linux distributions
- Installer support for different ROCm releases
- Installer support for specific ROCm components
- Optional driver or driver-only installer creation
- Optional post-install preferences
- Lightweight installer packages, which are unique to the preconfigured ROCm setup
- Resolution and inclusion of dependency packages for offline installation

2.2.5.1 Prerequisites

The ROCm Offline Installer Creator requires the following configuration:

- The host must be connected to the network and internet when running the ROCm Offline Installer Creator.
- The host system running the ROCm Offline Installer Creator and the target system running the installer must use the same Linux distribution, release version, and Linux kernel version.

For example, if the host system uses Ubuntu 22.04.4 with the 6.5.0-44-generic kernel, only an Ubuntu 22.04.4 target with the 6.5.0-44-generic kernel can use the offline installer. If the base OS distribution version and the kernel version of that distribution do not match on both systems, installation is not permitted.

The following configuration is recommended when using the ROCm Offline Installer Creator:

- The host system running the ROCm Offline Installer Creator and the target system should use the same OS image and have the same packages installed.
- The host OS should not have ROCm or the AMDGPU driver installed. (This recommendation doesn't apply to any AMDGPU drivers included with the default distribution packages.)

2.2.5.2 Supported Linux distributions

The ROCm Offline Installer Creator tool supports the following Linux distributions and versions:

- Ubuntu: 20.04, 22.04, 24.04
- RHEL: 8.10, 9.4, 9.5, 9.6
- SLES: 15.6
- Debian: 12

2.2.5.3 Getting started

The ROCm Offline Installer Creator is distributed as a self-extracting .run package. Launch the tool from any directory on the host system to create an offline installer.

Download the Offline Installer Creator from repo.radeon.com using the following command:

```
wget https://repo.radeon.com/rocm/installer/rocm-linux-install-offline/rocm-rel-<rocm-version>/
↪<distro>/<distro-version>/<creator-package>
```

Substitute your values for the following placeholders:

- <rocm-version>: ROCm version number for the ROCm Offline Installer Creator tool, for example, rocm-rel-6.4 or rocm-rel-6.4.1.
- <distro>: Linux distribution for the tool, for example, ubuntu, ol, rhel, sles, or debian.
- <distro-version>: Linux distribution version for the tool, for example, 22.04 for Ubuntu or 9.4 for RHEL.
- <creator-package>: The ROCm Offline Installer Creator package name, for example, rocm-offline-creator_1.0.0.60200-3~22.04.run.

> **Note**
>
> For releases that end in .0, do not include the .0 as part of the rocm-version component. For example, for ROCm 6.4.0, the rocm-version is rocm-rel-6.4.

For example, use this command to download ROCm version 6.4.1 of the Offline Installer Creator for Ubuntu release 22.04:

```
wget https://repo.radeon.com/rocm/installer/rocm-linux-install-offline/rocm-rel-6.4.1/ubuntu/22.04/rocm-
↪offline-creator_1.0.9.60401-3~22.04.run
```

2.2.5.4 Installer Creation

On the host system, run the ROCm Offline Installer Creator from the terminal command line as follows:

```
bash ./rocm-offline-creator_1.0.0-local.run <options>
```

The <options> parameter can either be left empty or set to these options:

- help: displays information on how to use the Offline Installer Creator
- version: displays the current version of the Offline Installer Creator
- prompt: enables user prompts during offline installer creation
- config=: specifies the full path to a .config file that contains a configuration for creating an offline installer. This parameter is used for testing purposes. See the Testing section for more information.
- wconfig=: specifies the full path to a .config file that is created using the current configuration settings when the user exits the user interface. When this option is specified, the ROCm Offline Installer Creator does not create an installer package. This option is used for testing purposes.

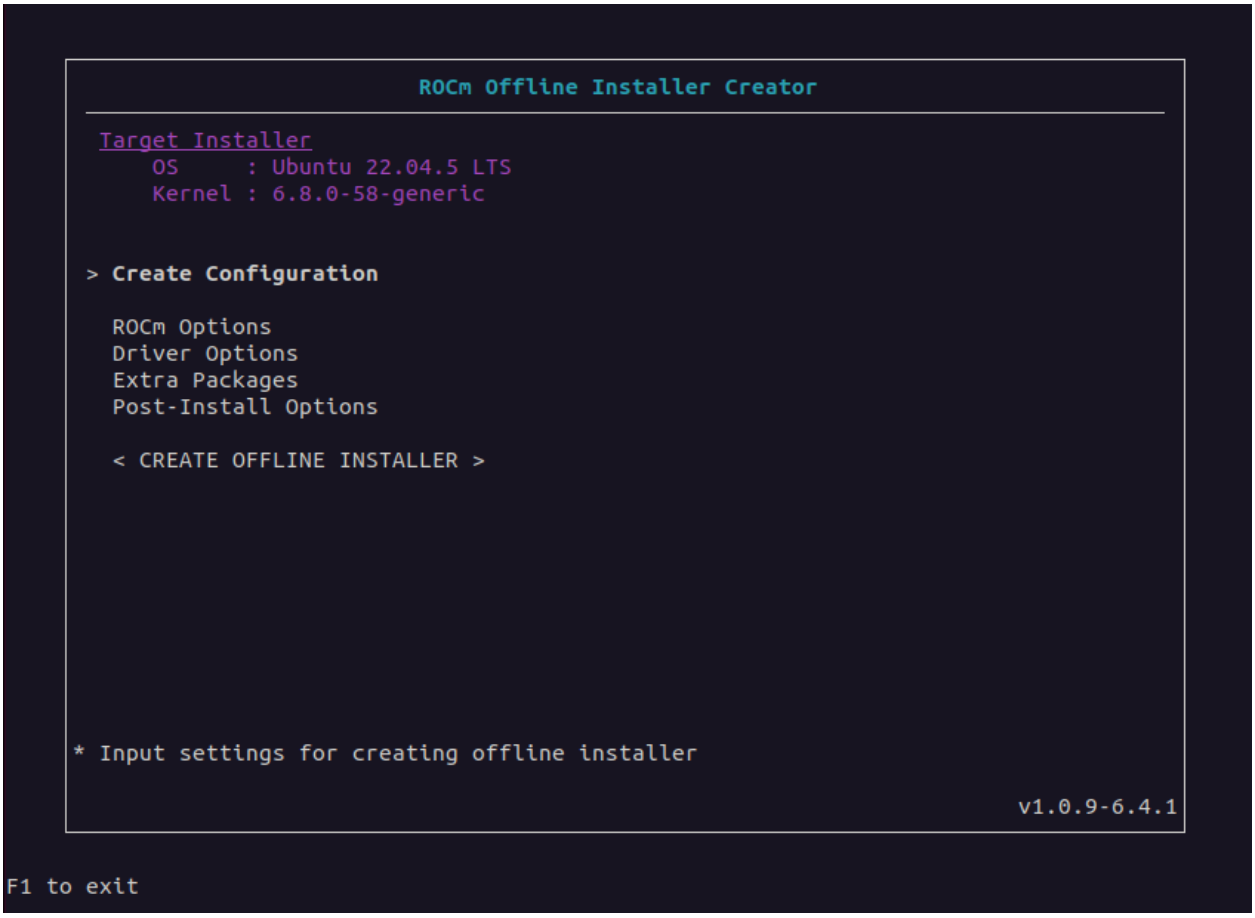
> **Note**
>
> The prompt and config options can be combined in the same command.

This example demonstrates how to use the prompt option when running the Offline Installer Creator:

```
bash ./rocm-offline-creator_1.0.0-local.run prompt
```

The optional prompt parameter stops the Offline Installer Creator at critical checkpoints in the creation process and prompts the user. At these checkpoints, the user can either continue or end the program. Without the prompt parameter, installer creation continues without interruption, unless a failure occurs during the process.

After extracting the .run package, the Offline Installer Creator displays the Main menu. From here, you can customize the offline installer using the Create Configuration menu and options in the ROCm, driver, and extra packages menus.



After configuring all the options, select Create Offline Installer to review the settings and start the offline installer creation process. After the installer is created, the offline installer .run package is saved to the configured location. Copy this package to any matching target system for ROCm and driver installation.

2.2.5.5 Offline Installer Creator interface

Use the Offline Installer Creator user interface to configure the contents of the ROCm and driver installation package for later use on the target system.

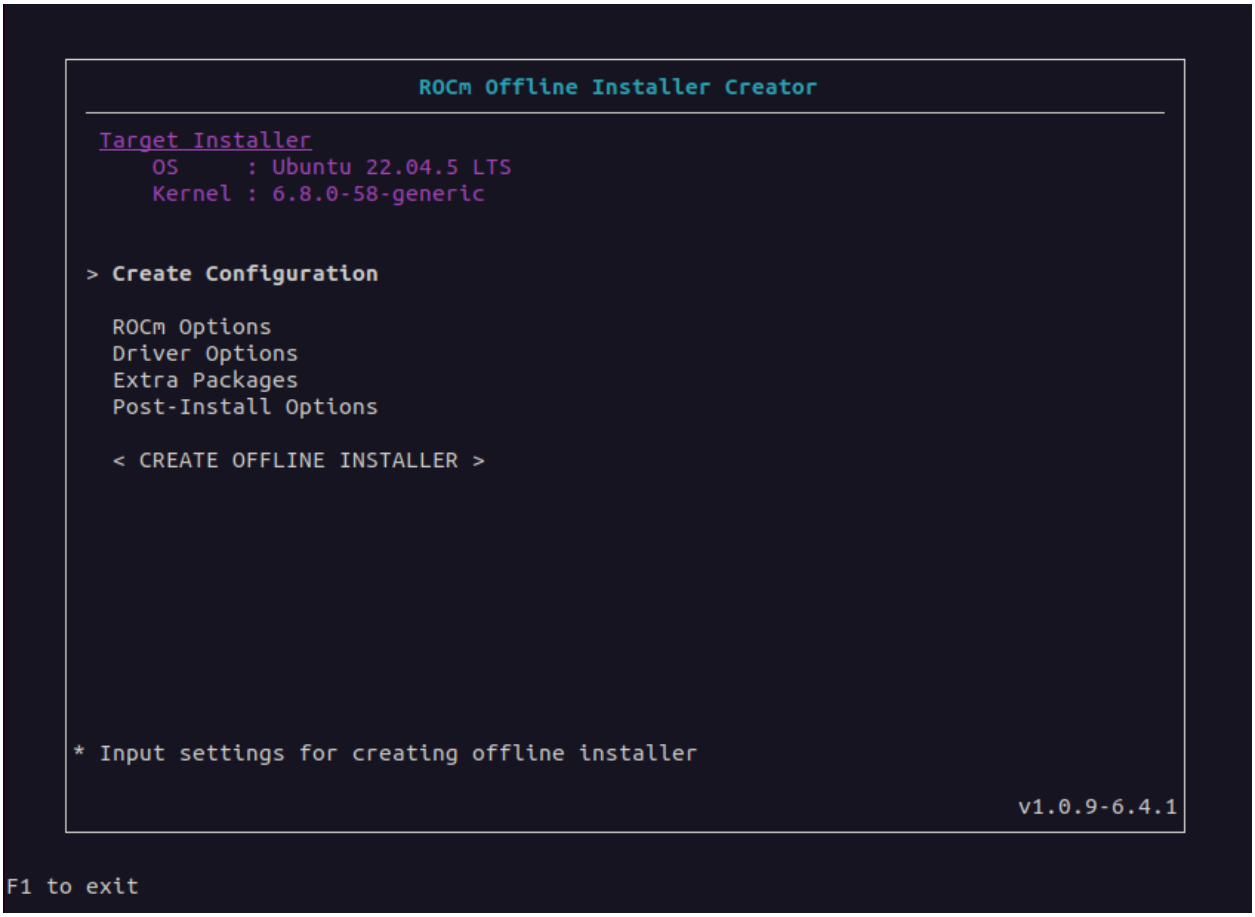
Starting from the Main menu, the Offline Installer Creator user interface contains multiple menus and sub-menus used to configure and create the offline installer. The following section describes the menu items:

- Main
- Create Configuration

- ROCm Options
- Driver Options
- Extra Packages
- Post-Install Options
- Create Offline Installer

2.2.5.5.1 Main menu

The Main menu is the starting point for installer configuration.



2.2.5.5.2 Create Configuration menu

The Create Configuration menu selects the input source and output parameters for the installer. Configuration items in this menu include the input repository, the type of dependency downloading to perform, and the name and output location for the new installer.

- Installer Input

  The Installer Input indicates the input repository for sourcing the ROCm and driver packages used to build the offline installer. A value of repo-public indicates that a publicly-available repository, as defined in the Repo Type field, is used for the package input.

- Repo Type

  The Repo Type indicates the specific type of repository used for input. If the value of Installer Input is set to repo-public, then repo.radeon.com is used to source and download any ROCm and driver packages required to create the offline installer.

- Dependency Download Type

  To create an offline installer, any package downloaded and installed on the target system must include any dependent packages required for offline installation. The Offline Installer Creator provides two options for controlling how dependent packages are included: full and minimum.

  Set the Dependency Download Type to full, the default setting, to resolve all dependencies for ROCm and the selected drivers aggressively and recursively and download them as required. Full mode typically results in a larger offline installer than minimum mode. This mode is recommended if the host system running the Offline Installer Creator uses a modified OS image with additional packages installed, such as ROCm.

  In cases where the host system is running an unmodified OS image that is exactly the same as the target, set the Dependency Download Type to minimum. This mode downloads the minimum number of dependent packages required to create the offline installer. The resulting offline installer has a smaller size in minimum mode than it does in full mode. A typical use of minimum mode is for distributing a

preconfigured OS image as the base installation across multiple offline systems. The base image is used to create an offline installer which is then deployed to the matching offline systems to install ROCm.

- Installer Name

  Installer Name specifies the name of the offline installer .run file that is generated. The default option is rocm-offline-install.

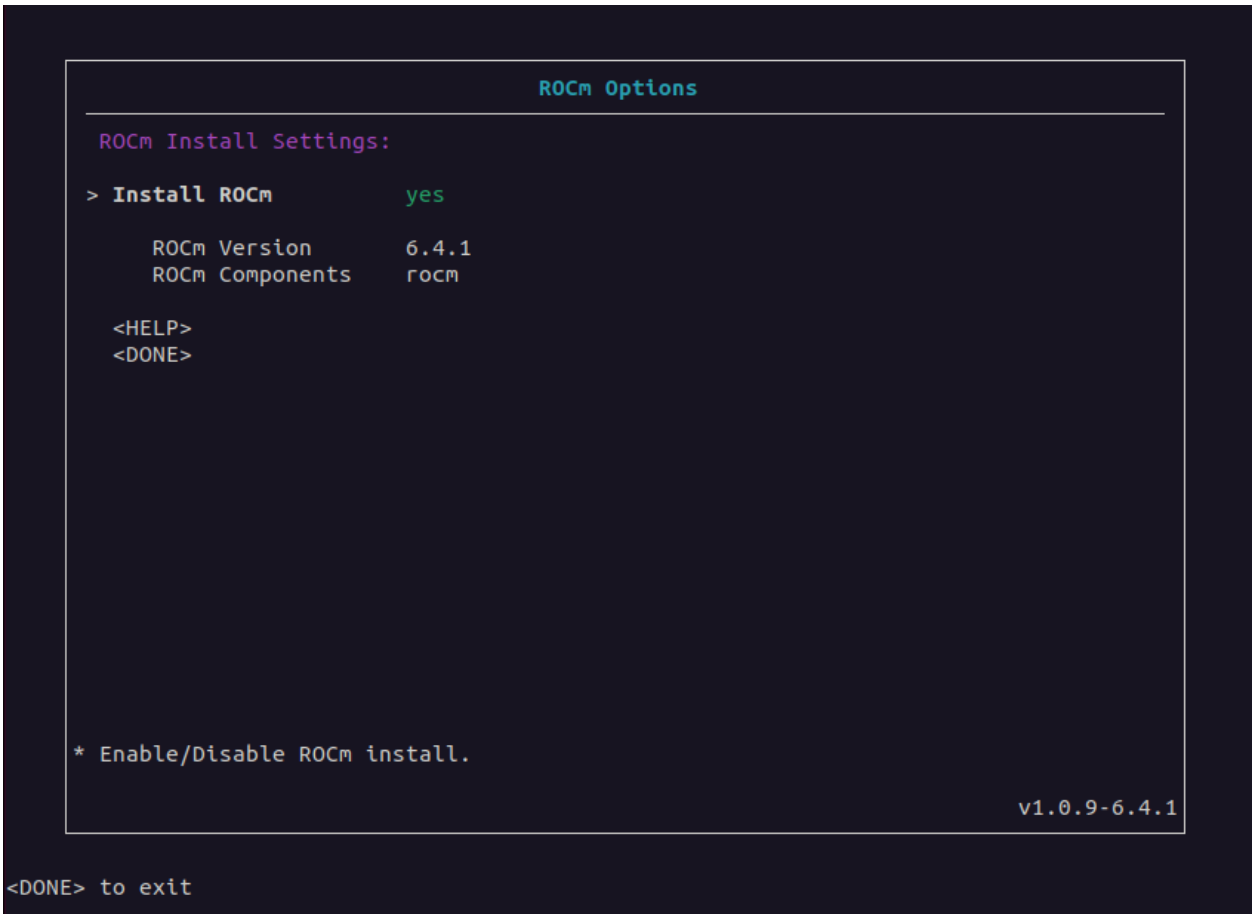
- Installer Path

  Installer Path configures the output directory for the new offline installer on the host system. The default location is the home directory of the current user.

> **Note**
>
> The Offline Installer Creator validates the directory path and doesn't permit installer creation if the path is invalid.

2.2.5.5.3 ROCm Options menu

The optional ROCm Options menu includes or excludes the ROCm component in the offline installer. If ROCm installation is included, a specific version and set of the ROCm components are integrated into the resulting installer.



- Install ROCm

This field indicates whether to include ROCm components in the offline installer. If this field is set to yes, ROCm installation is enabled. If you choose this configuration, you must select the ROCm version and list of ROCm components. Otherwise, the creation of the offline installer is not permitted. To create a "driver-only" offline installer, disable the Install ROCm option.

- ROCm Version

  If Install ROCm is enabled, select a specific version of ROCm using the ROCm Version sub-menu. ROCm version 5.7.3 and later are available for selection. All ROCm components are based on this version of ROCm.

  > **Note**
  >
  > Only one ROCm version can be selected.

- ROCm Components

  If Install ROCm is set to yes, you can select a list of components from the ROCm Components sub-menu. These components represent use cases and functions within the ROCm software stack. For example, rocm is one of the primary components within ROCm. It includes most of the essential parts of the ROCm stack, so most users should include rocm as a base component for any ROCm offline installer.

  > **Note**
  >
  > – Configure the ROCm Version field before selecting the ROCm components.
  >
  > – Select one or more components from the ROCm Components list for offline installer creation.

The following ROCm Components are available for offline installation. For more information on the components, see What is ROCm.

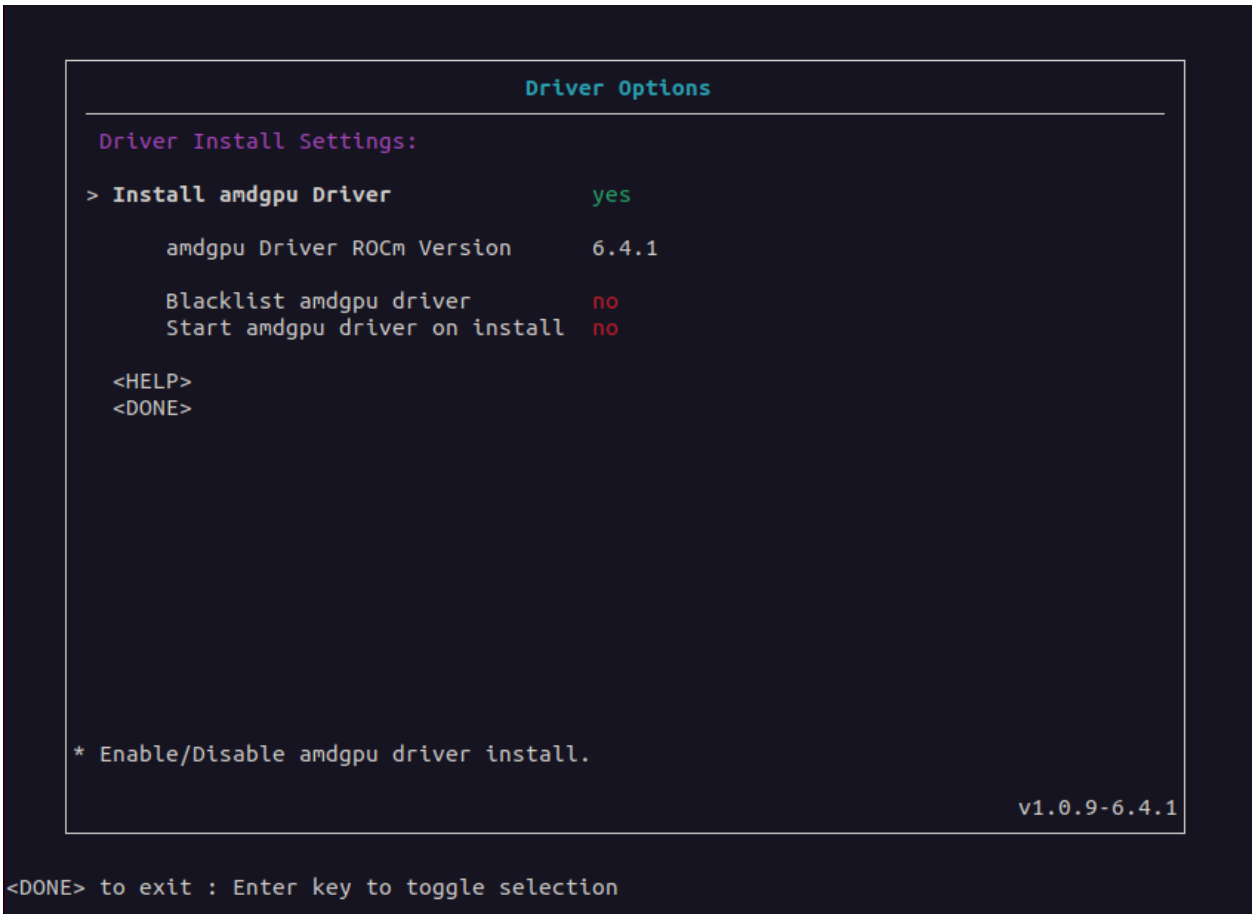
- rocm
  - For users and developers requiring the full ROCm stack
  - OpenCL (ROCr/KMD based) runtime
  - HIP runtimes
  - Machine-learning framework
  - All ROCm libraries and applications
- rocmdev
  - For developers requiring the ROCm runtime, with profiling and debugging tools
  - HIP runtimes
  - OpenCL runtime
  - Profiler, tracer, and debugger tools
- rocmdevtools
  - For developers requiring the ROCm profiling and debugging tools
  - Profiler, tracer, and debugger tools
- lrt

  - For users of applications using the ROCm runtime
  - ROCm compiler and device libraries
  - ROCr runtime and thunk
- hip
  - For users of the HIP runtime on AMD products
  - HIP runtimes
- hiplibsdk
  - For application developers using HIP on AMD products
  - HIP runtimes
  - ROCm math libraries
  - HIP development libraries
- graphics
  - For users of graphics applications
  - Open-source Mesa 3D graphics and multimedia libraries
- multimediasdk
  - For developers of open-source multimedia
  - Open-source Mesa 3D multimedia libraries
  - Development headers for multimedia libraries
- opencl
  - For users of applications requiring OpenCL on Vega or later products
  - ROCr-based OpenCL
  - ROCm language runtime
- openclsdk
  - For application developers requiring ROCr-based OpenCL
  - ROCr-based OpenCL
  - ROCm language runtime
  - Development and SDK files for ROCr-based OpenCL
- openmpsdk
  - For users of OpenMP or Flang on AMD products
  - OpenMP runtime and development packages
- mllib
  - For users running machine-learning workloads
  - MIOpen hip and tensile libraries
  - Clang OpenCL
  - MIOpen kernels
- mlsdk

- For developers running machine-learning workloads
- MIOpen development libraries
- Clang OpenCL development libraries
- MIOpen kernels

2.2.5.5.4 Driver Options menu

Use the Driver Options menu to optionally include the AMDGPU driver in the offline installer. If driver installation is included, an AMDGPU driver based on a specific ROCm version is integrated into the installer. In addition, the installer can configure several post-installation driver options for offline installation.



- Install amdgpu Driver

  This field specifies whether to include the AMDGPU driver in the offline installer. To enable its inclusion, set the value of Install amdgpu Driver to yes. To include the driver, select the amdgpu Driver ROCm Version. Otherwise, the creation of an offline installer is not permitted. This option is typically disabled when creating an ROCm-only offline installer.

- amdgpu Driver ROCm Version

  If Install amdgpu Driver is set to yes, the amdgpu Driver ROCm Version field is used to select a specific ROCm version to base the AMDGPU driver on.

> **Note**
>
> – The offline installer can only use one ROCm version.
>
> – The AMDGPU driver included in the offline installer is the DKMS version.

2.2.5.5.4.1 Post-install driver options

If you are including the AMDGPU driver in the offline installer, you can apply one or more of the following post-installation driver options:

- Blacklist amdgpu driver

  When this setting is enabled, the resulting offline installer immediately disables the AMDGPU driver kernel module after installing it. As a result, the system disables driver startup and initialization on subsequent reboots. Use this option when the AMDGPU driver is being debugged or is potentially unstable, or if control of driver initialization is required during boot up.

  > **Caution**
  >
  > This option is intended for advanced users familiar with Linux kernel modules and GPU drivers.

  > **Note**
  >
  > If this option is selected, the Start amdgpu driver on install option is not available.

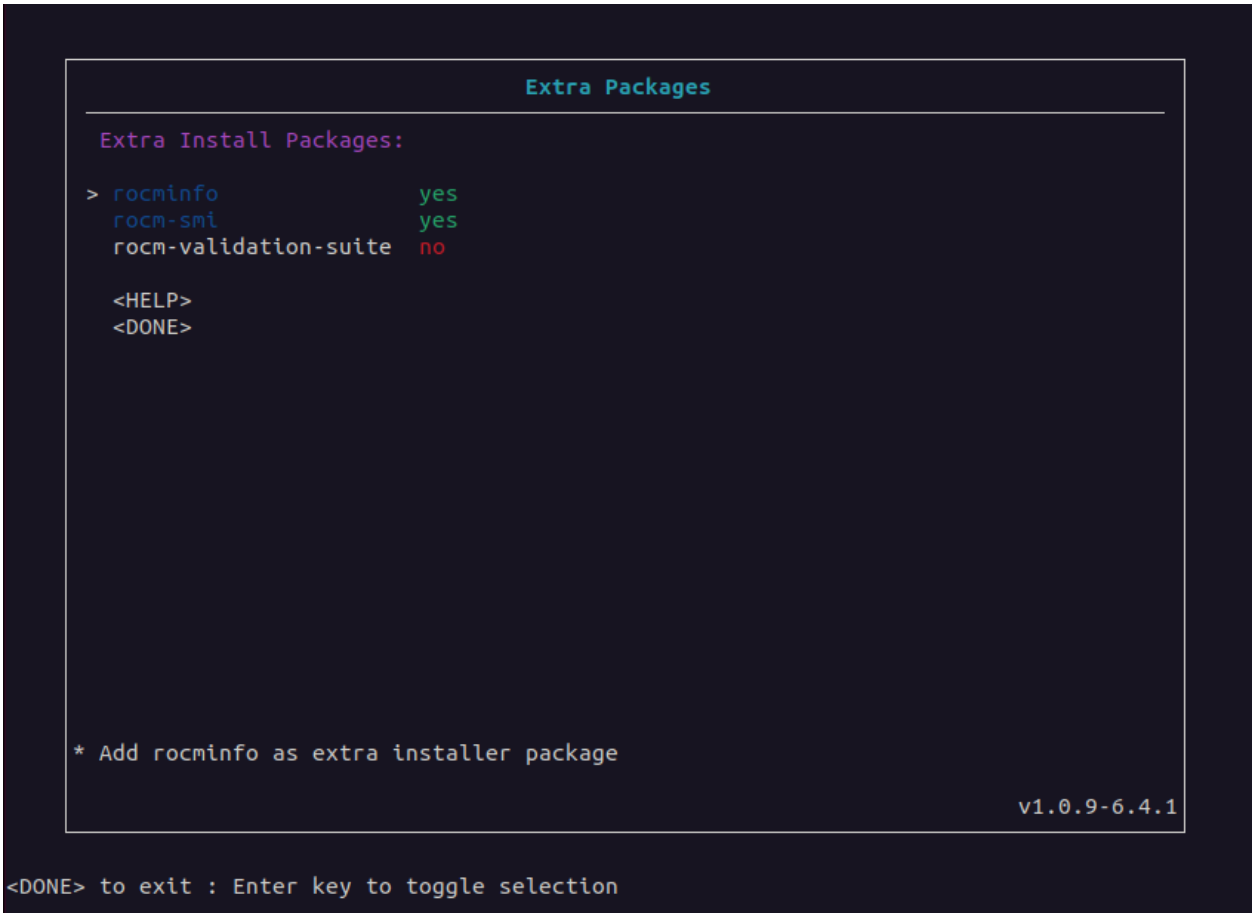
- Start amdgpu driver on install

  If the Start amdgpu driver on install option is enabled, the offline installer uses modprobe to automatically launch the AMDGPU driver after installation. If a pre-existing AMDGPU driver is already loaded on the system, the installer doesn't start the new driver. This option is often useful for users installing the driver on a system where the GPU device is newer and not yet natively supported as part of the upstream GPU driver for the Linux distribution.

  > **Note**
  >
  > If this option is selected, the Blacklist amdgpu driver on install option is not available for use in the offline installer.

2.2.5.5.5 Extra Packages menu

The Extra Packages menu provides a list of optional packages for inclusion in the offline installer. You can select the rocminfo, rocm-smi, and rocm-validation-suite packages as extra packages for the installer, provided they are not already included as part of a given ROCm component.

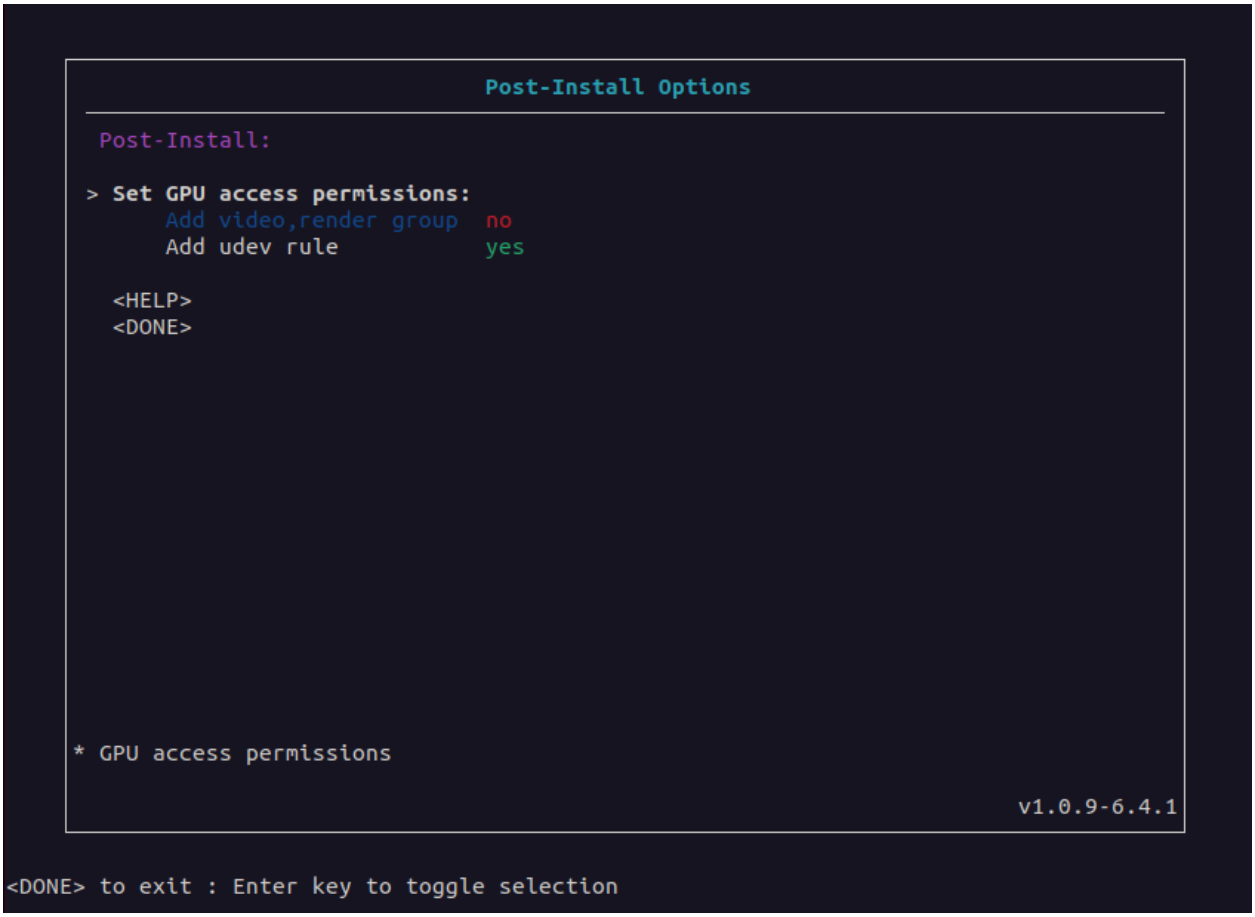


Note

If a selected ROCm component includes rocminfo or rocm-smi, the fields are set to yes and can't be modified.

2.2.5.5.6 Post-Install Options menu

Use the Post-Install Options menu to enable additional setup and configuration items after the initial ROCm installation.



- Set GPU access permissions

  This section configures the GPU access permissions after the ROCm installation. Typically, any ROCm component using the GPU and requiring access to GPU resources must set the GPU access permissions. For ROCm, GPU access is controlled by membership in the video and render groups. Group membership and access to GPU resources can be configured using one of these two methods.

  – Add video,render group

    If the Add video,render group field is enabled, the current user ($USER) is added to the groups and granted GPU access.

  – Add udev rule

    If the Add udev rule field is enabled, GPU access is granted to all users on the system.

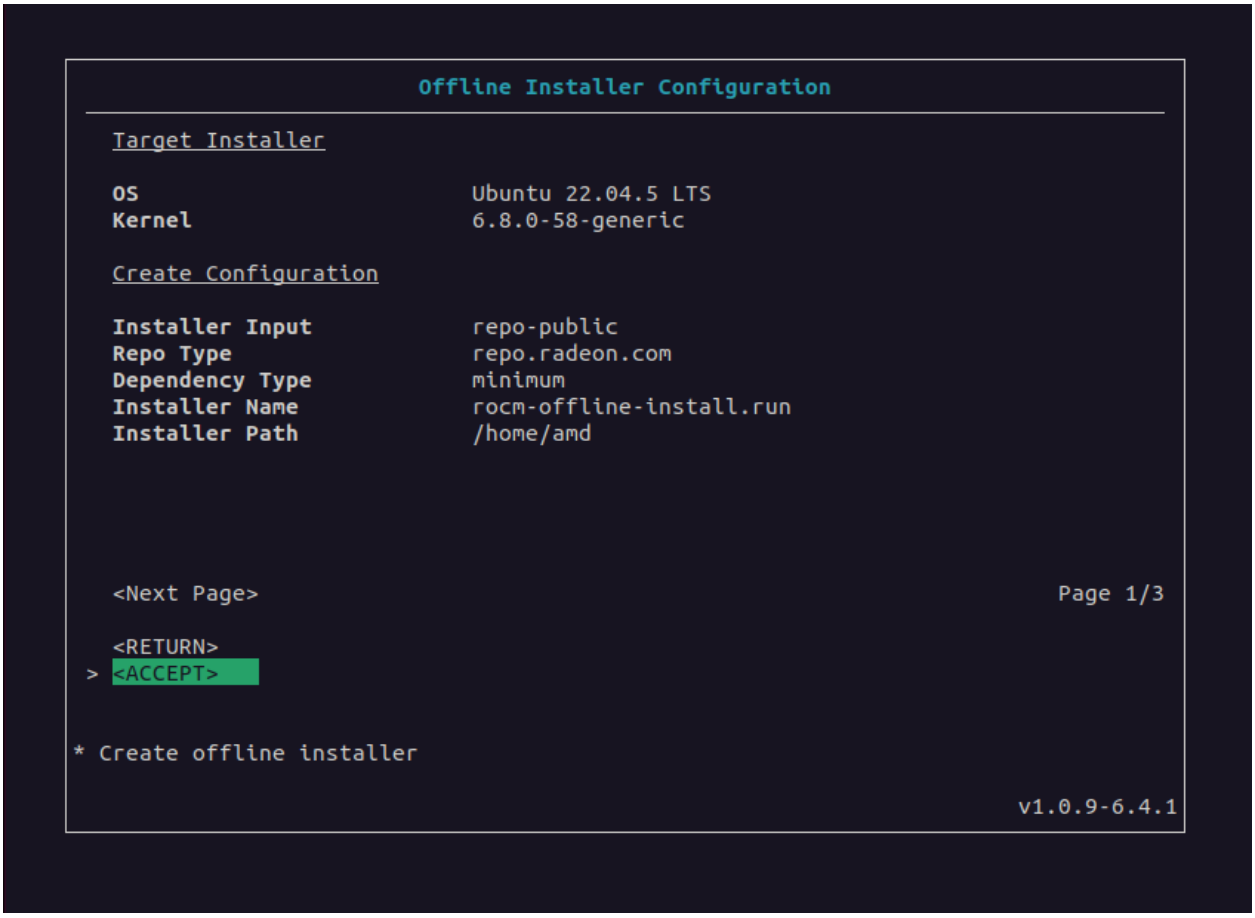
  > **Note**
  >
  > It's recommended that you enable one of the GPU access options to use ROCm. Only one method for adding GPU access permissions can be selected.

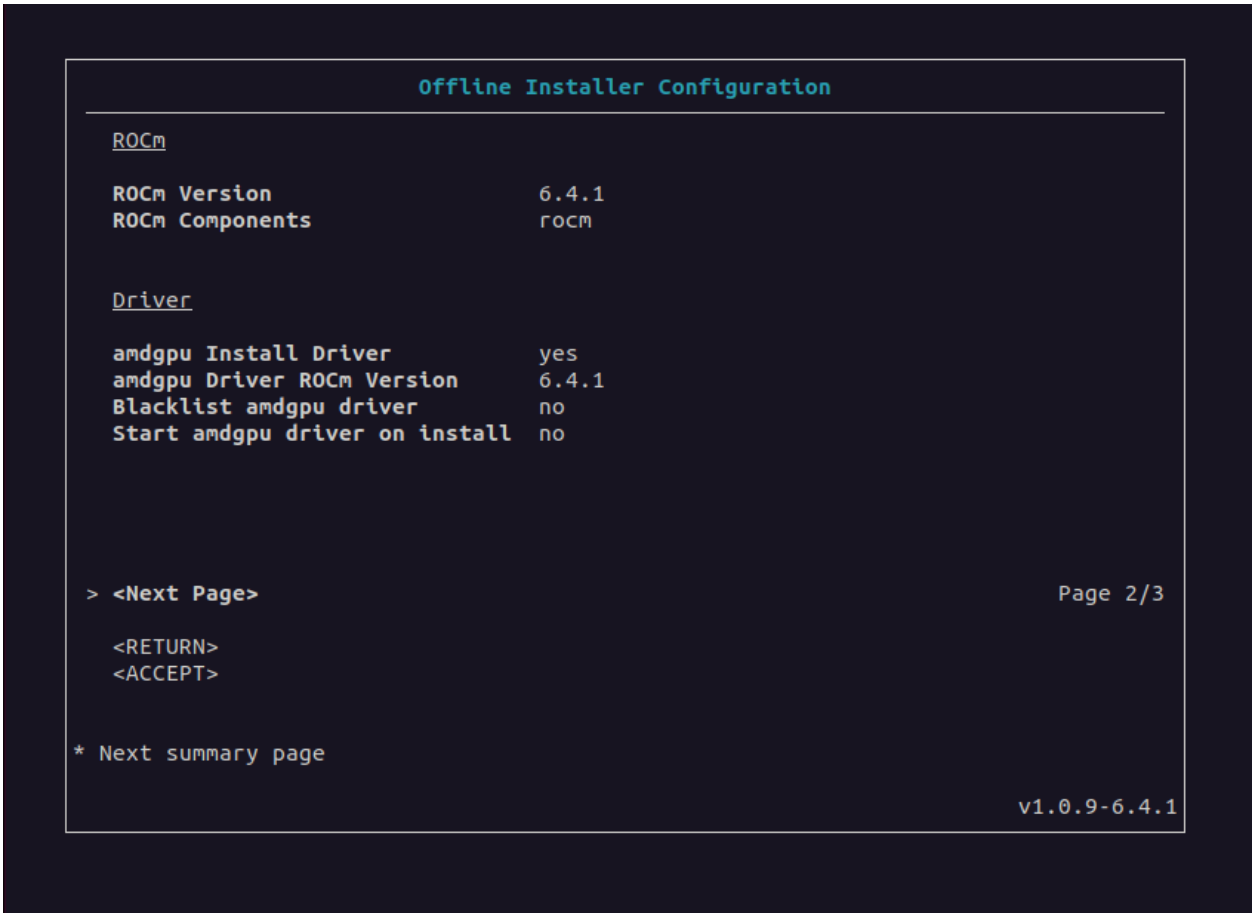
2.2.5.5.7 Create Offline Installer menu

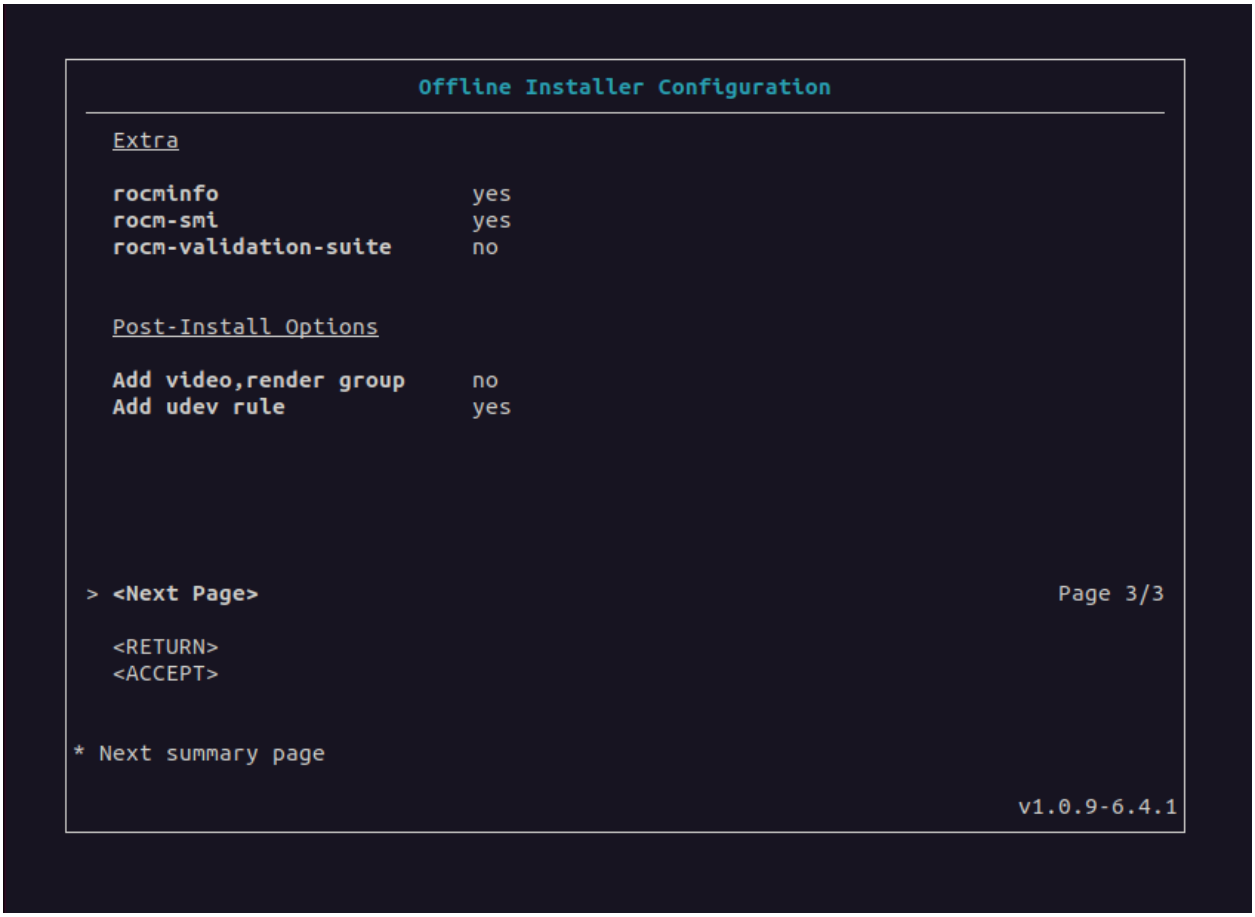
The Create Offline Installer menu summarizes the current configuration. The Offline Installer Creator uses this configuration to create the offline installer. If there are no errors, you can Accept the configuration and create the offline installer .run file. If the tool finds any missing requirements or errors in the configuration,

the Accept option is not available. In this case, return to the Main menu by selecting Return and edit the current configuration before proceeding.

The following illustrations show the three menu pages required to create the offline installer.




Offline Installer Configuration
ROCm
ROCm Version 6.4.1
ROCm Components rocm
Driver
amdgpu Install Driver yes
amdgpu Driver ROCm Version 6.4.1
Blacklist amdgpu driver no
Start amdgpu driver on install no
> <Next Page>
Page 2/3
<RETURN>
<ACCEPT>
* Next summary page
v1.0.9-6.4.1



2.2.5.6 Using the ROCm Offline Installer Creator

After starting the Offline Installer Creator, the user interface appears in the terminal. At the top of the Main menu, the Target Installer field indicates the Linux distribution, version, and Linux kernel version currently running on the host system. The target system must match the host system because the target installer is based on the current host distribution and kernel version. When navigating through the menus, you can use the Done option to return to the previous menu. Some menus have a Help option to display more information about the current menu.

Follow these steps to create an offline installer:

1. Fill out the configuration creation details:
    a. Enter the Create Configuration menu.
    b. Set the Dependency Download Type to either full, which is the default, or minimum.
    c. Set the Installer Name field or use the default name that is provided. The Offline Installer Creator appends the .run extension to the installer name when it creates the installer.
    d. Use the Installer Path field to configure the output directory for the installer. The default value is the $USER home directory.
2. Set the ROCm options:

    In the ROCm Options menu, enable or disable the installation of ROCm components. If ROCm installation is enabled, the ROCm Version and ROCm Components fields are also required.

    a. Enter the ROCm Options menu.

b. Set Install ROCm to yes to include ROCm components in the offline installer.

c. For the ROCm Version field, select the ROCm release version from the sub-menu.

d. For the ROCm Components field, select one or more ROCm components from the sub-menu.

3. Set the driver options:

   In the Driver Options menu, enable or disable the installation of the amdgpu driver. To enable driver installation, you must also configure the amdgpu Driver ROCm version field.

   a. Enter the Driver Options menu.

   b. Set Install amdgpu Driver to yes to include the amdgpu driver in the offline installer.

   c. If the amdgpu Driver ROCm Version field is not already populated, select the ROCm release version from the sub-menu.

   d. Configure the post-installation driver options, including Blacklist amdgpu driver and Start amdgpu driver on install.

   > **Note**
   >
   > The amdgpu Driver ROCm Version field is the same field as the ROCm Version field in the ROCm Options menu. You can set the ROCm release version from either menu.

4. Set the extra packages:

   In the Extra Packages menu, optionally include rocminfo, rocm-smi, and rocm-validation-suite in the offline installer.

   > **Note**
   >
   > Some ROCm components already include the rocminfo and rocm-smi packages, which are automatically added to the offline installer by default. If so, the user interface sets the rocminfo and rocm-smi fields to yes and ensures they can't be edited.

5. Set the post-install options:

   In the Post-Install Options menu, set the method of enabling GPU access permissions to Add video,render group or Add udev rule.

6. Create the installer:

   After completing the previous steps, you can create the offline installer.

   a. Enter the Create Offline Installer menu.

   b. Review the current installer configuration by using the review summary interface, clicking <Next Page> to cycle through the pages.

   c. If there are any errors or missing requirements in the configuration, press <RETURN> to display the main menu and make any required changes. Any errors or missing requirements are indicated in red.

   d. If there are no errors in the configuration, select <ACCEPT> to create the offline installer, which is saved to the designated location.

   e. At this point, the user interface closes and the offline install creation script starts running.

> **Note**
>
> You can only select <ACCEPT> if all configuration requirements are met and no errors are detected.

7. Offline Installation:

   After the offline installer has been created and saved to the designated output directory on the host system, copy the installer to any matching target system and run it. The target doesn't need to have a network or internet connection.

   a. Copy the offline installer, for example rocm-offline-install.run, to a target system that matches the host.

   b. Run the installer from the terminal command line:

   ```
   ./rocm-offline-install.run <option>
   ```

   The option parameter can be omitted or optionally set to prompt, dryrun, or uninstall. (See the section below for more information on the uninstall option.) The prompt option enables user prompts during the offline installation process. If the prompt parameter is appended, the installer halts at critical points during the installation process, prompting the user to continue or stop the installation. Without the prompt parameter, the offline installation runs until completion, unless a failure occurs during the process.

   To simulate the offline installation process, set the option parameter to dryrun. With dryrun, the installer outputs any errors it detects but does not install any packages.

   ```
   ./rocm-offline-install.run prompt
   ```

   or

   ```
   ./rocm-offline-install.run dryrun
   ```

   > **Note**
   >
   > You can run the installer with both the dryrun and prompt options. This simulates the offline installation and prompts you during the process.

8. Offline Uninstall:

   If ROCm is already installed on your system, you must first uninstall it before running the installation package. If you try to install ROCm when another version is installed on the system, the new installation will fail. If you are using the Ubuntu distribution, you can use the installation package you created to uninstall ROCm.

   To uninstall ROCm, run the installer from the terminal command line with the uninstall parameter:

   ```
   ./rocm-offline-install.run uninstall
   ```

   After the installation is complete, reboot your system. You can then run the offline installer to install the new version.

> **Note**
>
> The uninstall feature is only enabled for the Ubuntu distribution.

2.2.5.6.1 Log files

By default, the ROCm Offline Installer Creator records its output in log files in the /var/log/offline_creator directory. The output of both creation and installation processes is logged to this directory.

2.2.5.7 Building

Build the ROCm Offline Installer Creator from source using the files located at https://github.com/ROCm/rocm-install-on-linux/tree/develop/src/offline-installer.

To build the Offline Installer Creator:

1. Install the prerequisites

   You must install the following packages prior to building the Offline Installer Creator from source.

   Ubuntu

   ```
   sudo apt install cmake
   sudo apt install build-essential
   sudo apt install libncurses5-dev
   sudo apt install makeself
   sudo apt install wget
   ```

   Red Hat Enterprise Linux

   First, install the RHEL version-specific prerequisites:

   Install the following for RHEL 8.x:

   ```
   sudo dnf install wget
   wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
   sudo rpm -ivh epel-release-latest-8.noarch.rpm
   sudo crb enable
   ```

   Install the following for RHEL 9.x:

   ```
   sudo dnf install wget
   wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.rpm
   sudo rpm -ivh epel-release-latest-9.noarch.rpm
   sudo crb enable
   ```

   Then install the generic RHEL x.x prerequisites

   ```
   sudo dnf install cmake
   sudo dnf install gcc gcc-c++
   sudo dnf install ncurses-devel
   sudo dnf install makeself
   ```

SUSE Linux Enterprise Server

Install the following for SLES 15.5:

```
SUSEConnect -p PackageHub/15.5/x86_64
sudo zypper install cmake
sudo zypper install gcc gcc-c++
sudo zypper install ncurses-devel
sudo zypper install makeself
```

Install the following for SLES 15.6:

```
SUSEConnect -p PackageHub/15.6/x86_64
sudo zypper install cmake
sudo zypper install gcc gcc-c++
sudo zypper install ncurses-devel
sudo zypper install makeself
```

2. Clone the tool source

```
git clone https://github.com/ROCm/rocm-install-on-linux
```

3. Build the configuration

```
cd rocm-install-on-linux/src/offline-installer
mkdir build
cd build
cmake ..
```

4. Build the Offline Installer Creator

   The Offline Installer Creator is built for both standard and test use. The output is placed in the CMake build directory.

   - To build the tool, run the following command:

     ```
     make
     ```

     This command generates the file rocm-offline-creator_1.0.0-local.run.

2.2.5.8 Testing

The Offline Installer Creator test suite includes a set of tests for validating common installer configurations. These tests simulate and validate the creation of the target offline installer and the installer's behavior.

The tests are based on the ROCm version and on the components being installed.

Tests are available for these ROCm versions:

- 5.7.3
- 6.0.2
- 6.1.x
- 6.2.x
- 6.3.x
- 6.4.x

Tests are available for the following component combinations:

- ROCm only: Creates an installer for the rocm component only.
- Driver only: Creates an installer for the amdgpu driver only.
- ROCm and driver: Creates an installer for both the rocm component and the amdgpu driver.
- ROCm and graphics: Creates an installer for the rocm,graphics component.
- hip and hiplibsdk: Creates an installer for the hip,hiplibsdk component.

2.2.5.8.1 Building the tests

Tests are built using CMake and make. See the Building section above.

2.2.5.8.2 Running the tests

Run the preconfigured tests using the CMake and CTest utilities.

> **Note**
>
> Run any of the tests below with the -V argument to enable verbose logging and output for the offline installer tool creator or installer scripts.

There are two test types:

- Full test
- ROCm version test

For each type of test, installer creation is validated using one of the preset .config files located in the tests directory. These files contain the settings required to create the offline installer. They run through the create-offline.sh script for the relevant Linux distribution. This is followed by a dry run of the resulting offline installer .run file, which performs a simulated installation for the distribution under test.

2.2.5.8.2.1 Full test

From the build location of the offline tool, run the following command:

```
ctest
```

This suite runs 111 tests.

The following tests are available, depending on the ROCm version:

ROCm version	Test Suite Support
5.7.3	ROCm only, Driver only, ROCm + Driver, ROCm + graphics, hip + hiplibsdk
6.0.2	ROCm only, Driver only, ROCm + Driver, ROCm + graphics, hip + hiplibsdk
6.1.x	ROCm only, Driver only, ROCm + Driver, ROCm + graphics, hip + hiplibsdk
6.2.x	ROCm only, Driver only, ROCm + Driver, ROCm + graphics, hip + hiplibsdk
6.3.x	ROCm only, Driver only, ROCm + Driver, ROCm + graphics, hip + hiplibsdk
6.4.x	ROCm only, Driver only, ROCm + Driver, ROCm + graphics, hip + hiplibsdk

> **Note**
>
> Test suites for a ROCm version only run if the version is supported on the Linux Distribution Version under test.

2.2.5.8.2.2 ROCm version test

Select subsets of the "Full" test for a specific version of ROCm and run them using CTest.

From the build location of the offline tool, run the following command:

```
ctest -L <rocm-version>
```

where <rocm-version> is one of 5.7.3, 6.0.2, 6.1.x, 6.2.x, 6.3.x, or 6.4.x.

2.2.5.8.2.3 Running manual tests

In addition to the preconfigured CTest utilities, you can manually configure your own Offline Installer Creator tests. This configuration option bypasses the user interface and runs the create-offline.sh script as described in the Running the tests section above. To manually test the creation of an offline installer without the user interface, run the Offline Installer Creator with the config parameter.

> **Caution**
>
> For normal Offline Installer Creator usage, the user interface is recommended. The user interface validates the various configuration file inputs and guides you through the configuration process.

```
bash ./rocm-offline-creator_1.0.0-local.run config=[path-to-config-file]
```

Set the config parameter to the absolute path to a configuration file. The file name must include the .config file extension. The file follows the format defined by the Offline Installer Creator user interface. Example configuration files, which are used for running the CTest utilities, are available in the tests directory of the offline-installer repository.

2.2.6 ROCm Runfile Installer

The ROCm Runfile Installer installs ROCm, the AMDGPU driver, or a combination of the two on a system with or without network or internet access. Unlike all other installation methods, the ROCm Runfile Installer can install ROCm and the AMDGPU driver without using a native Linux package management system.

The key advantage of using the ROCm Runfile Installer is its offline installation support. Many system environments have network or internet access restrictions, making installation via normal package management difficult. Furthermore, some installation environments might also have general restrictions on package management usage. Therefore, the ROCm Runfile Installer lets you perform a completely self-contained ROCm software installation.

The ROCm Runfile Installer includes these features:

- An optional easy-to-use user interface for configuring the installation
- An optional command line interface for the installation
- Offline ROCm and AMDGPU driver installation (requires the prior installation of dependencies)
- Packageless ROCm and AMDGPU driver install without native package management

- A single self-contained installer for all ROCm and AMDGPU driver software
- Configurable installation location for the ROCm install

2.2.6.1 Prerequisites

The ROCm Runfile Installer requires the following configuration:

- Installation of dependency requirements for the ROCm runtime
- Installation of dependency requirements for the AMDGPU driver (optional)
- Sufficient storage space for the installation (100 GB of free space)
- A supported Linux distribution

2.2.6.2 Dependency requirements

The ROCm components contained within the ROCm Runfile installer have a specific set of libraries, frameworks, and other elements that must be pre-installed on the system before you can use ROCm after the installation. Similarly, the inclusion of the AMDGPU driver as part of the installation also requires specific libraries that must be pre-installed. ROCm Runfile installer users must pre-install the list of required first-level dependencies.

To install the pre-install dependencies, use one of two methods:

- Manual installation
- The ROCm Runfile Installer

2.2.6.2.1 Manual installation

You can determine the dependent packages from the ROCm Runfile Installer Pre-Install Configuration Settings menu in the GUI or from the command line by using the deps=list rocm argument for ROCm or the deps=list amdgpu argument for the AMDGPU driver. This list indicates all packages required for the ROCm runtime or the AMDGPU driver. The required libraries, frameworks, and other components within the required packages must be on the system when running ROCm or installing the AMDGPU driver. Users can manually install the required packages in the list using any method.

System administrators might prefer a manual installation process when deploying ROCm across a multi-node cluster environment, where a base operating system image is prepared and applied. This base OS image might have the dependency requirements pre-installed.

2.2.6.2.2 ROCm Runfile Installer

For single-system environments, users can choose to have the ROCm Runfile Installer automatically install the dependency requirements as part of the pre-installation stage for ROCm or the AMDGPU driver. Any missing dependency requirements can be installed using the Install Dependencies option of the Pre-Install Configuration Settings menu in the GUI or from the command line using the deps=install rocm or deps=install amdgpu argument.

> **Note**
>
> The ROCm Runfile Installer requires a network or internet connection to install the dependency requirements.

2.2.6.3 Supported Linux distributions

The ROCm Runfile Installer tool supports the following Linux distributions and versions:

- Ubuntu: 22.04, 24.04
- RHEL: 8.10, 9.4, 9.5, 9.6
- SLES: 15.6

2.2.6.4 Getting started

The ROCm Runfile Installer is distributed as a self-extracting .run file. To install ROCm, launch the installer from any directory on the system.

2.2.6.4.1 Downloading the ROCm Runfile Installer

Download the ROCm Runfile Installer from repo.radeon.com using the following command:

```
wget https://repo.radeon.com/rocm/installer/rocm-runfile-installer/rocm-rel-<rocm-version>/<distro>/
↪<distro-version>/<installer-file>
```

Substitute values specific to your installation for the following placeholders:

```
<rocm-version>    = ROCm version number for the installer
<distro>          = Linux distribution for the installer
<distro-version>  = Linux distribution version for the installer
<install-file>    = The installer .run file
```

For example, use this command to download ROCm version 6.4.1 of the ROCm Runfile Installer for Ubuntu release 22.04:

```
wget https://repo.radeon.com/rocm/installer/rocm-runfile-installer/rocm-rel-6.4.1/ubuntu/22.04/rocm-
↪installer_1.1.1.60401-30-83~22.04.run
```

2.2.6.4.2 Running the ROCm Runfile Installer

After downloading the ROCm Runfile Installer, run it from a terminal using GUI install or Command line install. See the sections below for more details.

You can obtain help or version information using the following installer .run file argument options:

```
bash rocm-installer.run help
bash rocm-installer.run version
```

> **Note**
>
> These commands use rocm-installer.run as a placeholder for the actual run file. Throughout the guide, substitute the name of the actual .run file for rocm-installer.run.

Both the help and version commands run without extracting the installer contents. Depending on the install method, they provide quick feedback on how to use the installer. For all other argument options, or if no arguments are specified, the installer .run file self-extracts to the current working directory where the .run file is executing. The self-extraction process creates a new directory named rocm-install containing the content and tools required for the installation. The rocm-install directory also includes a logs directory for recording the installation process. For more information, see Log files below.

> **Note**
>
> The installer self-extraction process might take a significant amount of time due to the size of the installer content and the decompression process.

2.2.6.5 Install methods

The ROCm Runfile Installer provides two methods for running the ROCm installation:

- GUI install: The GUI installation includes a visual interface for configuring the installation, letting you specify the pre- and post-installation requirements. In addition, the GUI provides feedback and guidance for setting up the installation. This method is recommended for new and intermediate installer users.
- Command line install: The command line interface installation method provides a direct terminal-based approach for configuring and running the installation. This method is recommended for more advanced installer users.

2.2.6.6 GUI install

Launch the GUI-based installation of the ROCm Runfile Installer from the terminal command line without arguments as follows:

```
bash rocm-installer.run
```

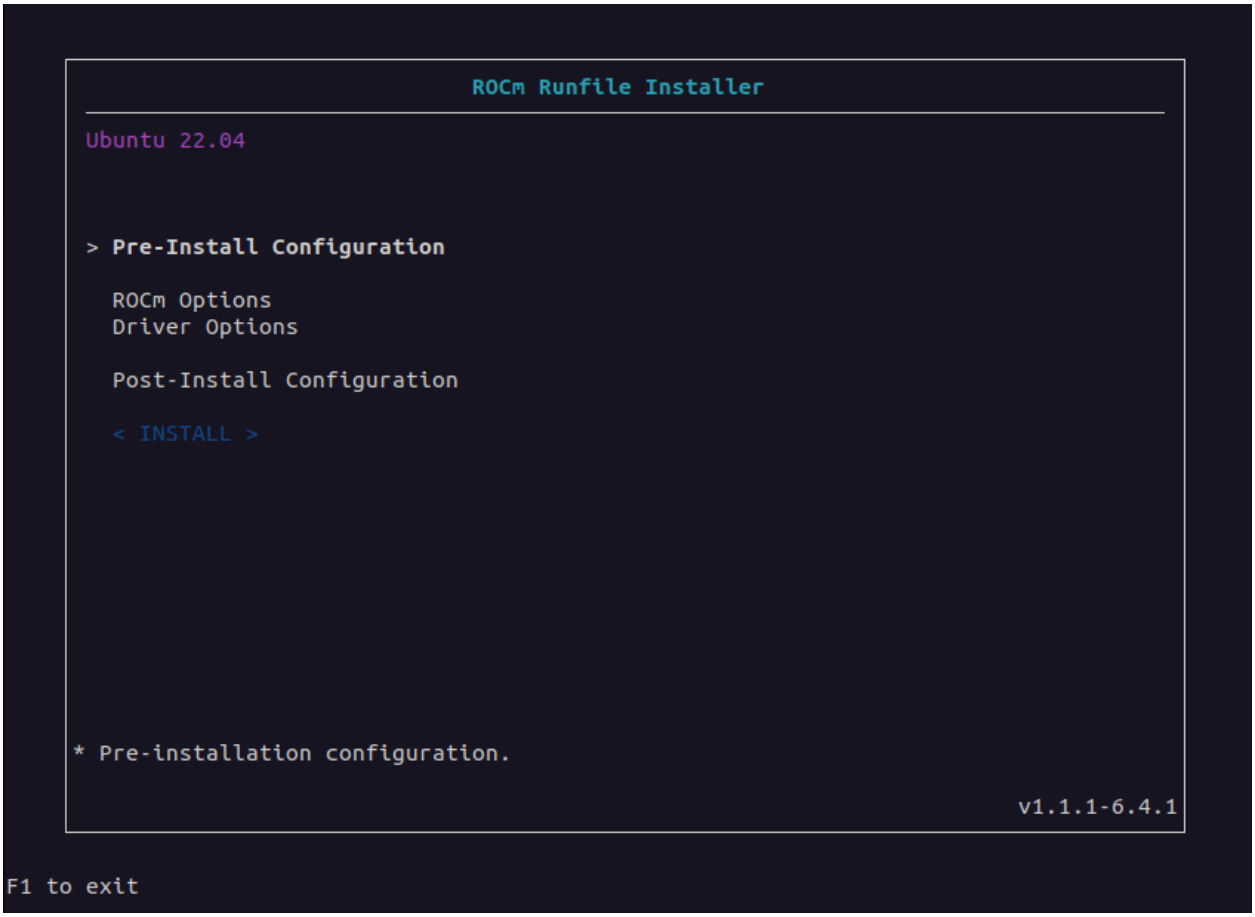
2.2.6.6.1 GUI

Use the Runfile Installer GUI to configure the installation, from the pre- to post-install options.

Starting from the Main menu, the user interface contains multiple menus and sub-menus for each stage of the installation process.

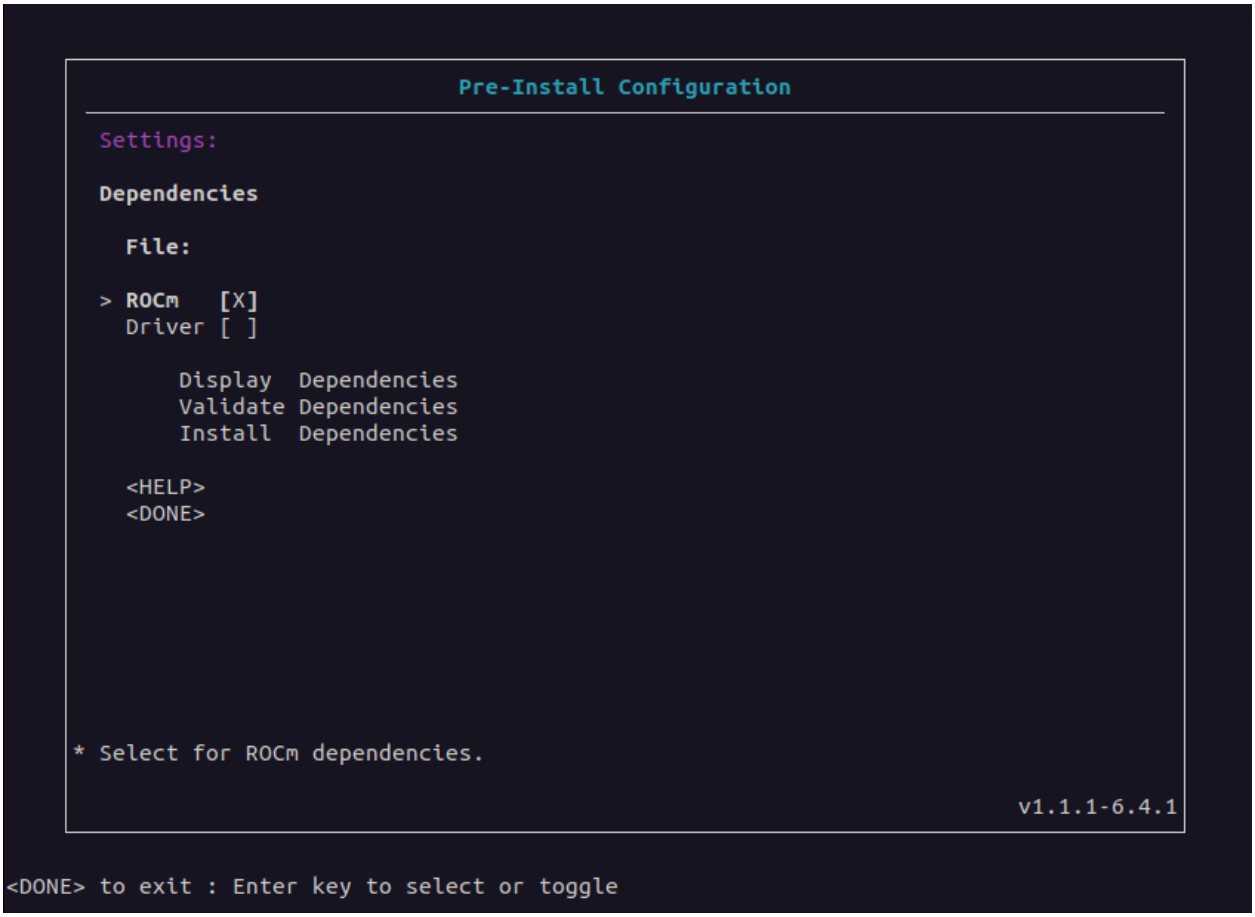
2.2.6.6.1.1 Main menu

The Main menu is the installation starting point.



2.2.6.6.1.2 Pre-Install Configuration Settings menu

The Pre-Install Configuration Settings menu is an optional menu used to configure pre-installation requirements before installation. The pre-installation settings relate to the dependent libraries and packages required by the ROCm runtime or the AMDGPU driver.



- File

  File displays the location of the deps_list.txt file. This file is based on the combination of selected dependencies using the ROCm and Driver checkboxes and either Display Dependencies or Validate Dependencies. If Display Dependencies is selected, deps_list.txt includes a list of all required dependencies. If Validate Dependencies is selected, deps_list.txt contains only the missing dependencies on the system that still require installation. The File field is initially blank until Display Dependencies or Validate Dependencies is selected.

- ROCm / Driver

  The ROCm and Driver checkboxes are used to select which dependencies you want to display, validate, or install.

- Display Dependencies

  Display Dependencies lists all required (Debian or RPM) packages that must be pre-installed on the system. These packages are required by ROCm or the AMDGPU driver being installed by a particular ROCm Runfile Installer version.

  > **Note**
  >
  > The required packages are listed in the deps_list.txt file and can be installed separately from the ROCm Runfile Installer.

- Validate Dependencies

Validate Dependencies verifies which required packages are currently installed on the system where ROCm or the AMDGPU driver is being installed. It displays which packages from the required packages list are missing.

> **Note**
>
> The missing packages are listed in the deps_list.txt file.

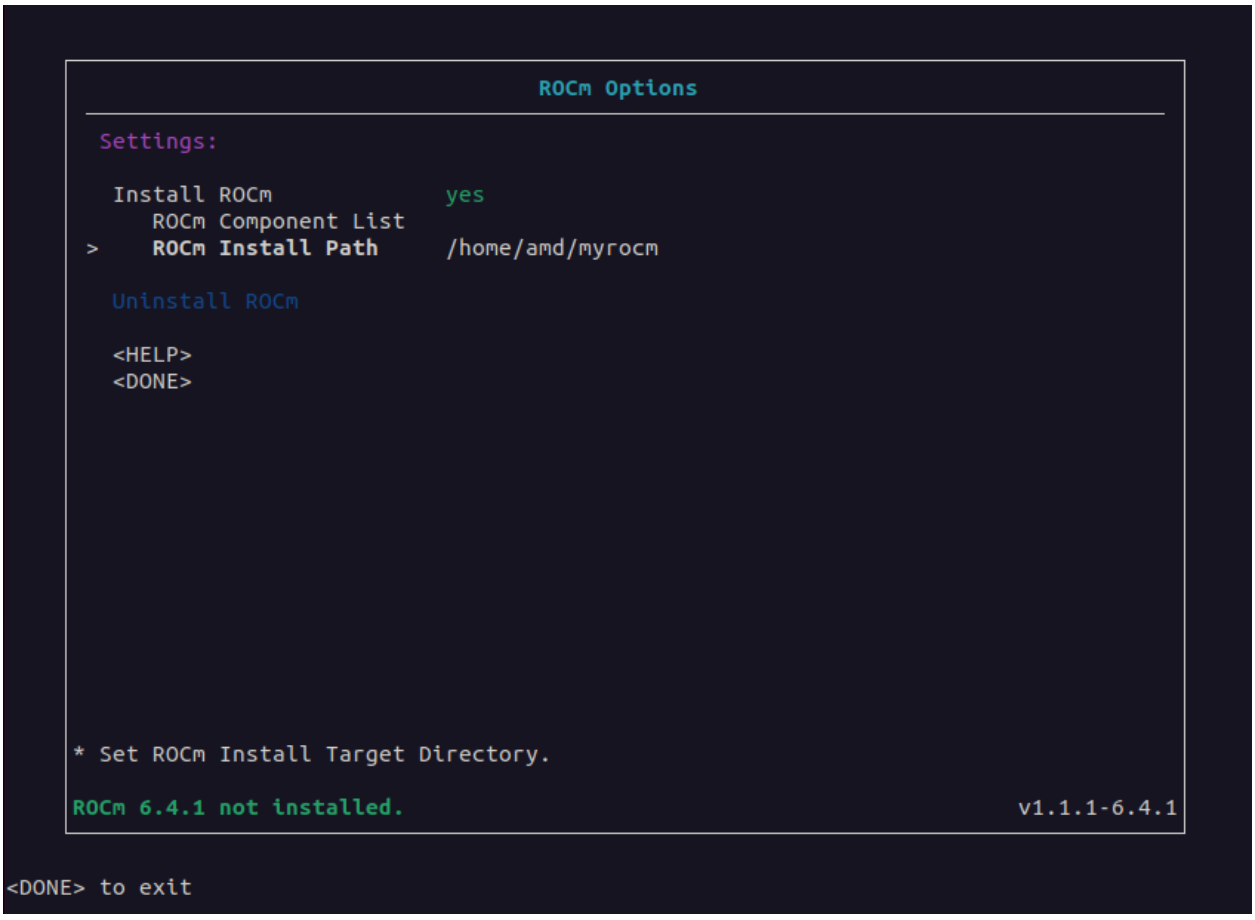
- Install Dependencies

  If the installer is running on the system where ROCm or the AMDGPU driver will be installed, you can choose to install any missing dependencies using the Install Dependencies option.

> **Note**
>
> Install Dependencies is only intended for a system with network or internet access. For offline installation using the ROCm Runfile Installer, you must manually install the required packages. For more details, see the Dependency requirements section.

2.2.6.6.1.3 ROCm Options menu

The ROCm Options menu can include or exclude ROCm from the installation.



- Install ROCm

This field indicates whether to include ROCm components in the installation. If this field is set to yes, ROCm installation is enabled and the Runfile Installer searches the system for any existing ROCm installations. If a previous ROCm installation is detected, the Uninstall ROCm field becomes selectable.

- ROCm Component List

  If Install ROCm is set to yes, this field displays a list of all ROCm components and component versions included in the installation.

- ROCm Install Path

  If Install ROCm is set to yes, the ROCm Install Path field can set the full path to the directory where ROCm will be installed. The default location is /, which is the typical /opt/rocm ROCm installation location.

  When ROCm Install Path is set to a new path, the installer validates the new directory location. If the directory exists, the ROCm installation can proceed. If an invalid location is specified, the ROCm installation will not be allowed.

- Uninstall ROCm

  This field is only available if previous Runfile ROCm installations are discovered on the system where ROCm is being installed, based on the currently selected ROCm Install Path location. The installer only lists previous installation locations that are on the current install path. If any installation locations are present on this path, they can be selected for uninstall. The installer indicates the type of installation as follows:

  – P (Package manager): Package manager installation that matches the ROCm version for the Runfile Installer. In this case, uninstall is not allowed.

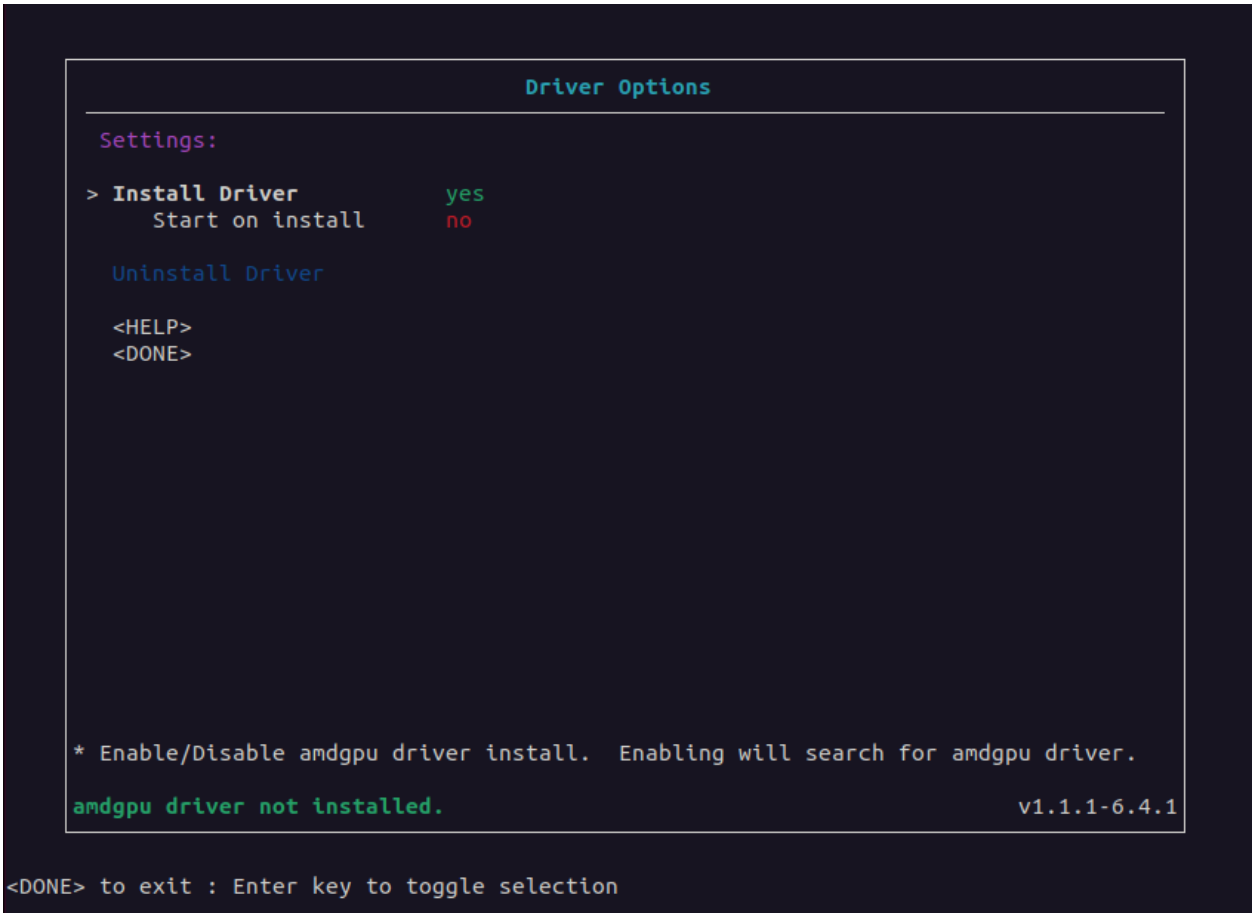
  – C (Runfile conflict): Runfile installation that matches the ROCm version for the Runfile Installer. The conflicting installation can be uninstalled.

  – R (Runfile): Runfile installation that differs from the ROCm version for the Runfile Installer. The installation can be uninstalled.

  Uninstall ROCm is for a single ROCm instance at a time and is only available for Runfile installs. Package manager installs of ROCm cannot be uninstalled by the Runfile installer and must be uninstalled manually using the package management application.

2.2.6.6.1.4 Driver Options menu

The Driver Options menu can include or exclude the AMDGPU driver from the installation.



- Install Driver

  This field indicates whether to include the AMDGPU driver in the installation. If this field is set to yes, AMDGPU driver installation is enabled. When this field is enabled, the system is searched for any existing AMDGPU driver installations. If a previous AMDGPU driver installation is detected, the Uninstall Driver field becomes selectable.

- Start on install

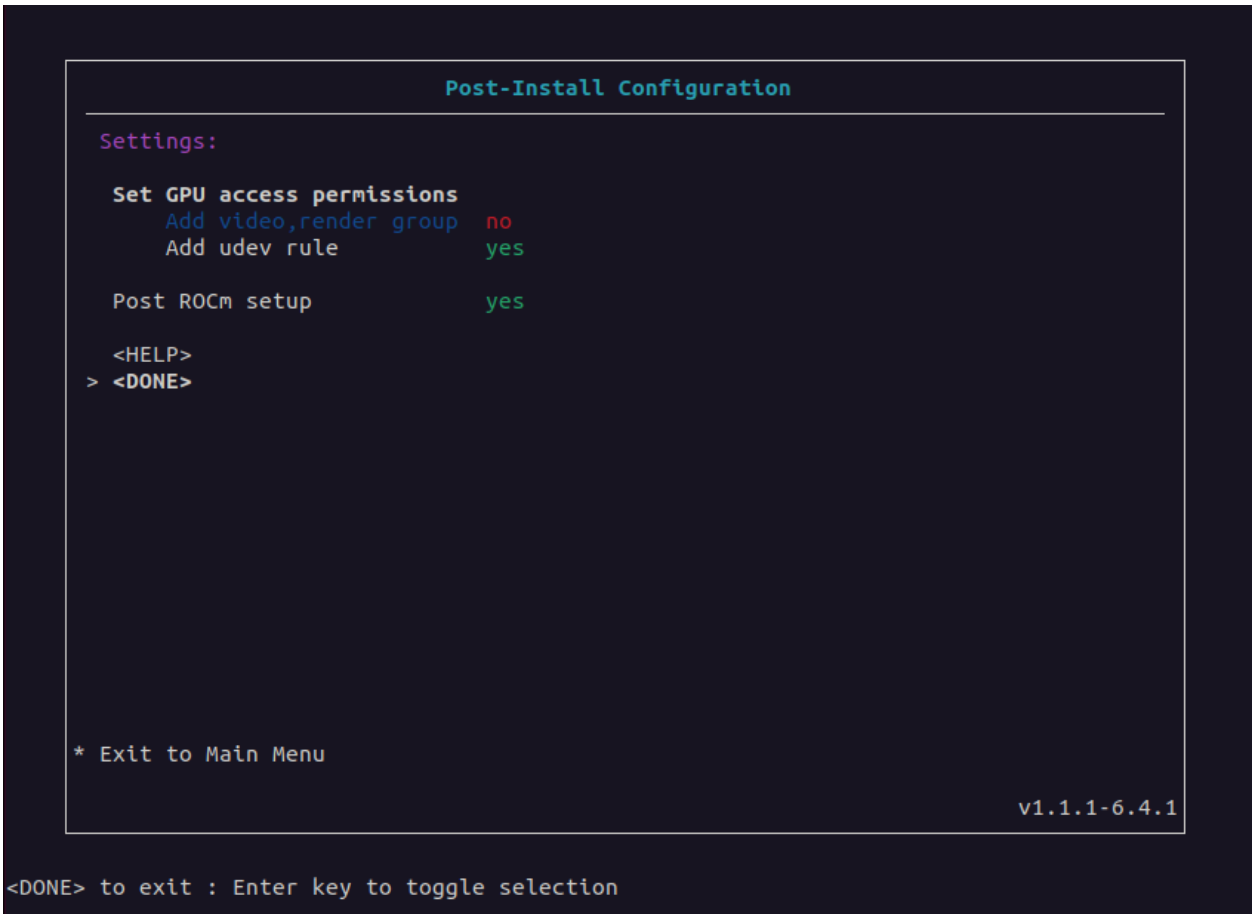
  If the Start on install option is enabled, the Runfile installer uses modprobe to automatically launch the AMDGPU driver after installation. If a pre-existing AMDGPU driver is already loaded on the system, the new driver will not start. This option can be useful for installing the driver on a system where the GPU device is newer and not yet natively supported as part of the upstream GPU driver for the Linux distribution.

- Uninstall Driver

  This field is only available if a previous Runfile install of the AMDGPU driver is discovered on the install system. If a Runfile installation of the AMDGPU driver is detected, select Uninstall Driver to remove it. Package manager installs of the AMDGPU driver cannot be uninstalled by the Runfile installer and must be uninstalled manually using the package management application.

2.2.6.6.1.5 Post-Install Options menu

Use the Post-Install Options menu to optionally enable additional setup and configuration items after the initial ROCm install.



- Set GPU access permissions

  This section sets the GPU access permissions after the ROCm installation. Typically, any ROCm component using the GPU and requiring access to GPU resources needs to set the access permission. For ROCm, GPU access is controlled by membership in the video and render groups. Membership and access to GPU resources can be set using one of two methods.

  – Add video,render group

    If the Add video,render group field is enabled, the current $USER is added to the groups and will be granted GPU access.

  – Add udev rule

    If the Add udev rule field is enabled, GPU access is granted to all users on the system.

  > **Note**
  >
  > It's recommended that you enable one of the GPU access options for using ROCm.
  >
  > Only one method for adding GPU access can be selected.

- Post ROCm setup

  When this option is enabled, the ROCm post-install setup is performed. This includes configuring symbolic links and other system requirements for using ROCm and the ROCm runtime.

  > **Tip**
  >
  > It's recommended that you enable the Post ROCm setup to guarantee proper functioning of the ROCm components and applications. However, advanced users who understand the ROCm setup might want to disable this option so they can control the post-installation ROCm setup for their specific environment.

2.2.6.6.2 Using the GUI

Start the ROCm Runfile Installer user interface from the terminal and launch the Main menu. While navigating through the user interface menus, use the Done option to return to the previous menu. Some menus have a Help option to display more information about the elements within the current menu.

Follow these steps to install ROCm and the AMDGPU driver:

1. (Optional) Install dependencies:
   a. Enter the Pre-Install Configuration menu.
   b. Select the ROCm checkbox, Driver checkbox, or both to specify the dependency type.
   c. If using the installer to install missing required dependencies, select Install Dependencies.

      To manually install the required dependencies, select Display Dependencies to list all the dependencies or Validate Dependencies to only list the missing dependencies on the current system. Quit the installer using DONE->F1 and separately install the required dependencies, which will be listed in the deps_list.txt file at the File location. After completing this task, restart the ROCm Runfile Installer and proceed to step 2.
2. Set the ROCm options:
   a. Enter the ROCm Options menu.
   b. Set Install ROCm to yes to include ROCm components in the installation.
   c. (Optional) Select Uninstall ROCm to uninstall a previous Runfile installation.
   d. Leave the ROCm Install Path field set to the default location to install ROCm to /opt/rocm or set the install location to a valid existing directory.
3. Set the AMDGPU driver options:
   a. Enter the Driver Options menu.
   b. Set Install Driver to yes to include the AMDGPU driver in the installation.
   c. (Optional) Select Start on install to load the driver after installation.
   d. (Optional) Select Uninstall Driver to uninstall a previous Runfile installation.
4. Set the post-install options:
   a. Set the method of enabling permission for GPU access to Add video,render group or Add udev rule.
   b. Set Post ROCm setup to yes to include post-install ROCm setup configuration.

2.2.6.7 Command line install

The command line install interface can be used as an alternative to the menu-based ROCm Runfile Installer GUI to reduce user interaction during the installation.

Run the ROCm Runfile Installer from the terminal command line as follows:

```
bash rocm-installer.run <options>
```

The <options> parameter can be set to these options:

- User help/information
  - help: Displays information on how to use the ROCm Runfile Installer.
  - version: Displays the current version of the ROCm Runfile Installer.
- Runfile options
  - noexec: Disable all installer execution. Extract the .run file content only.
  - noexec-cleanup: Disable cleanup after installer execution. Keep all .run extracted and runtime files.
- Dependencies
  - deps=<arg> <compo>:
    - <arg>:
      - list <compo>: Lists the required dependencies for the install <compo>.
      - validate <compo>: Validates which required dependencies are installed or not installed for <compo>.
      - install-only <compo>: Installs the required dependencies only for <compo>.
      - install <compo>: Installs with the required dependencies for <compo>.
      - file <file-path>: Installs with the dependencies from a dependency configuration file with path <file_path>.
      - file-only <file-path>: Install the dependencies from a dependency configuration file with path <file_path> only.
    - <compo>: Install component (rocm/amdgpu/rocm amdgpu).
- Install
  - rocm: Enable ROCm components install.
  - amdgpu: Enable AMDGPU driver install.
  - force: Force the ROCm and AMDGPU driver install.
  - target=<directory>: The target directory path for the ROCm components install.
- Post-install
  - postrocm: Run the post-installation ROCm configuration (for instance, script execution and symbolic link creation).
  - amdgpu-start: Start the AMDGPU driver after the install.
  - gpu-access=<access_type>
    - <access_type>:
      - user: Adds the current user to the render,video group for GPU access.

        - all: Grants GPU access to all users on the system using udev rules.

- Uninstall
  - uninstall-rocm (target=<directory>): Uninstall ROCm.
    - (target=<directory>): Optional target directory for the ROCm uninstall.
  - uninstall-amdgpu: Uninstall the AMDGPU driver.
- Information/Debug
  - findrocm: Search for a ROCm installation.
  - complist: List the version of ROCm components included in the installer.
  - prompt: Run the installer with user prompts.
  - verbose: Run the installer with verbose logging.

> **Note**
>
> The installer can be used with multiple <options> combinations to enable specific stages of the ROCm install process (pre-installation and post-installation). The exception is any option that uses the keyword -only will apply that option only and no others. Some informational options are also single-option commands. These options are described in more detail below.

This example demonstrates how to perform a typical ROCm installation on a single target system with the required dependencies installed, the ROCm install directory set to /myrocm, GPU access set to udev, and with the post-install setup:

```
bash rocm-installer.run deps=install target="/myrocm" rocm gpu-access=all postrocm
```

This example demonstrates how to perform a typical AMDGPU driver installation on a single target system with the required dependencies installed:

```
bash rocm-installer.run deps=install amdgpu
```

Finally, you can combine the ROCm and AMDGPU driver installations:

```
bash rocm-installer.run deps=install target="/myrocm" rocm amdgpu gpu-access=all postrocm
```

2.2.6.7.1 Command line interface

The command line interface for the ROCm Runfile Installer is based on the <options> list provided to the installer .run file.

2.2.6.7.1.1 Runfile options

The ROCm Runfile Installer is a self-extracting .run file with a few options for controlling the extraction process. When the installer .run file starts execution, the extraction process begins with a checksum validation to verify the integrity of the .run file. If there are no errors, it begins extracting and decompressing the package contents. The contents are output to a new directory named rocm-installer, located in the current working directory.

Usually, when extraction and decompression are complete, execution begins automatically by either starting up the GUI (if no arguments are provided) or executing the rocm-installer.sh script with all command line arguments. The rocm-installer.sh script is in the extracted rocm-installer directory.

When the GUI exits or the command line completes execution, the Runfile installer will automatically clean up the rocm-installer directory and delete all content except for the log files and the deps_list.txt file.

In some cases, you might want to execute multiple commands from the same extraction and save the time required to verify the checksum and extract and decompress the package contents of the .run file. Two command line options let you disable the .run cleanup process: noexec and noexec-cleanup.

- noexec

  The noexec option is a single command line argument that lets the checksum and extraction process complete and then exits without starting the GUI or executing the rocm-installer.sh script. All content will be maintained after the exit. You can then use the rocm-installer.sh script directly from the command line without specifying the .run file name.

  For example, extract the .run file and then use rocm-installer.sh instead of rocm-installer.run to install ROCm and the AMDGPU driver separately:

```
bash rocm-installer.run noexec
cd rocm-installer
bash rocm-installer.sh rocm
bash rocm-installer.sh amdgpu
```

  > **Note**
  >
  > If the noexec option is used, all other <options> on the command line will be ignored.

- noexec-cleanup

  The noexec-cleanup option disables the cleanup process after the GUI or command line interface exits. Unlike the noexec option, all command line arguments are processed as normal, but no content is deleted upon exit or completion. At this point, you can switch to using the rocm-installer.sh script within the rocm-installer directory to avoid re-extracting the contents.

2.2.6.7.1.2 Dependency options

Like the GUI Pre-Install Configuration menu, the command line interface can list, validate, and install the required dependencies. At the command line, add the deps=<arg> <compo> option to the list of <options> for the .run file. The <compo> option can include any combination of the rocm and amdgpu tags.

- deps=list <rocm/amdgpu>

  This dependency option lists all the dependencies required for the .run file-based ROCm installation, AMDGPU installation, or both. It lists all required (Debian or RPM) packages that require pre-installation on the system. The additional rocm or amdgpu parameter is a requirement for the deps=list option that instructs the installer to list only the dependencies required by ROCm, the AMDGPU driver, or both.

  Running deps=list causes the installer to quit after listing the dependencies. deps=list can combine rocm and amdgpu and output the combined list of required dependencies.

  Use deps=list <rocm/amdgpu> as a single <options> parameter:

```
bash rocm-installer.run deps=list rocm
bash rocm-installer.run deps=list amdgpu
bash rocm-installer.run deps=list rocm amdgpu
```

> **Note**
>
> The list of required packages can be installed separately from the Runfile Installer.

- deps=validate <rocm/amdgpu>

  This dependency option verifies whether any of the required ROCm or AMDGPU driver packages in the dependency list are already installed on the system running the command. The output is a list of any missing dependency packages that require installation.

  Running deps=validate causes the installer to quit after listing the missing dependencies. deps=validate can combine rocm and amdgpu and output the combined list of missing dependencies.

  Use deps=validate <rocm/amdgpu> as a single <options> parameter:

```
bash rocm-installer.run deps=validate rocm
bash rocm-installer.run deps=validate amdgpu
bash rocm-installer.run deps=validate rocm amdgpu
```

> **Note**
>
> The list of missing packages can be installed separately from the Runfile Installer.

- deps=install

  This dependency option validates and installs any required packages in the dependency list for ROCm, the AMDGPU driver, or both that are missing on the system running the command. This dependency option is not a single <options> parameter and can be added to a list of other options for the installer. The deps=install option expects at least one of the rocm or amdgpu <options> parameters to also be present in the list to enable the pre-installation of required dependencies before the Runfile Installer installs ROCm, the AMDGPU driver, or both.

  For example, to install the dependencies and ROCm, the command line is as follows:

```
bash rocm-installer.run deps=install rocm
```

  To install the dependencies and the AMDGPU driver, the command line is as follows:

```
bash rocm-installer.run deps=install amdgpu
```

  To install the dependencies and both ROCm and the AMDGPU driver, the command line is as follows:

```
bash rocm-installer.run deps=install rocm amdgpu
```

> **Note**
>
> The installer can be set to only install the ROCm dependencies, AMDGPU dependencies, or both and then quit. In this case, add -only to the deps=install option:
>
> ```
> bash rocm-installer.run deps=install-only rocm
> bash rocm-installer.run deps=install-only amdgpu
> bash rocm-installer.run deps=install-only rocm amdgpu
> ```

- deps=file <file-path>

  This dependency option specifies the name of a input file for the installer. This file contains a custom list of dependency packages to install. The list must have as the same format as the output of the deps=list option. Specify each (Debian or RPM) package by name, one package per line. <file-path> is the second parameter, which indicates the absolute path to the dependency file.

  For example, to install the dependencies listed in a file named mydeps.txt as part of a ROCm install, the command line is as follows:

```
bash rocm-installer.run deps=file /home/amd/mydeps.txt
```

  > **Note**
  >
  > The installer can be set only to install the dependencies file and then quit. In this case, add -only to deps=file.
  >
  > ```
  > bash rocm-installer.run deps=file-only /home/amd/mydeps.txt
  > ```

2.2.6.7.1.3 Install options

Like the GUI ROCm Options menu, the ROCm Runfile Installer can be configured to install ROCm, which can be installed in a specific location.

- rocm

  At the command line, add the rocm option to enable ROCm installation.

  > **Note**
  >
  > This option must be in the list of .run <options> for ROCm component installation.

  For example, to install ROCm with no other options, the command line is as follows:

```
bash rocm-installer.run rocm
```

- target=<directory>

  This install option is used to set the target directory where ROCm will be installed. The target option is only used as an option for ROCm installation and is not required for an AMDGPU driver install.

  When target=<directory> is not specified in the <options> list, the installer uses a default installation path for ROCm. For the command line interface method, the default install directory is $PWD. This is the current working directory where you launched the installer. In this configuration, the installer creates a new directory inside $PWD named rocm and installs all ROCm components to this location.

  The user can change the default location and set the ROCm component installation directory using the target=<directory> option. The <directory> argument must be a valid and absolute path to a directory on the system executing the ROCm Runfile Installer.

  For example, to install ROCm to the usual /opt/rocm location, the command line is as follows:

```
bash rocm-installer.run target="/" rocm
```

  To install ROCm to a directory called amd/myrocm in the $USER directory, the command line is as follows:

```
bash rocm-installer.run target="/home/amd/myrocm rocm"
```

- amdgpu

  At the command line, add the amdgpu option to enable AMDGPU driver installation.

  > **Note**
  >
  > This option must be in the list of .run <options> for AMDGPU driver installation.

  For example, to install the AMDGPU driver with no other options, the command line is as follows:

```
bash rocm-installer.run amdgpu
```

  > **Note**
  >
  > Both rocm and amdgpu can be combined in the <options> list to install both components. For this case, the command line is as follows:
  >
  > ```
  > bash rocm-installer.run rocm amdgpu
  > ```

- force

  The force install option can be added to the <options> list for the case of multiple pre-existing Runfile installs of ROCm. Add this option to disable any installer prompts that ask for confirmation to continue with the ROCm install if there are currently one or more Runfile ROCm installations already on the system.

2.2.6.7.1.4 Post-install options

Similar to the GUI Post-Install Options menu, the post-install options can configure the ROCm Runfile Installer to apply additional post-installation options after completing the installation. At the command line, add one or more of the post-installation options to the <options> list for the .run file.

- postrocm

  This post-install option applies any post-installation configuration settings for ROCm following installation on the target system. The post-installation configuration includes any symbolic link creation, library configuration, and script execution required to use the ROCm runtime.

  To enable the ROCm post-install configuration, add postrocm to the <options> list at the command line.

  For example, to enable ROCm post-installation configuration for a ROCm installation to the / directory, the command line is as follows:

```
bash rocm-installer.run target="/" rocm postrocm
```

  If the postrocm option was not included as part of the initial ROCm install command, the post-installation task can still be run separately after the installation. To run the ROCm post-installation operation from the command line, use the postrocm argument in conjunction with target=rocm-install-path, where rocm-install-path is the location of the Runfile-installed ROCm installation. The ROCm installation version and the Runfile Installer version must match. For example,

if the current Runfile Installer is for ROCm 6.4.1, then rocm-install-path must indicate the path to a ROCm 6.4.1 Runfile installation.

To use the postrocm argument separately from the initial install of ROCm 6.4.1 to /home/amd/myrocm, run:

```
bash rocm-installer.run target="/home/amd/myrocm/rocm-6.4.1" postrocm
```

> **Note**
>
> Adding the postrocm option to the <options> list is highly recommended to guarantee proper functioning of the ROCm components and applications.

- gpu-access=<access_type>

  This post-install option sets the GPU resource access permissions. ROCm runtime libraries and applications might need access to the GPU. This requires setting the access permission to the video and render groups using the <access_type>.

  If the ROCm installation is for a single user, then set the <access_type> for the gpu-access option to user.

  For example, to add the current user ($USER) to the video,render group for GPU access for a ROCm installation, the command line is as follows:

```
bash rocm-installer.run rocm gpu-access=user
```

  In cases where a system administrator is installing ROCm for multiple users, they might want to enable GPU access permission for all users. For this case, set the <access_type> for the gpu-access option to all:

```
bash rocm-installer.run rocm gpu-access=all
```

> **Note**
>
> Adding the gpu-access option to the <options> list is recommended for using ROCm. A typical ROCm installation includes both the postrocm option and one of the gpu-access types.

2.2.6.7.1.5 Uninstall options

These options configure the ROCm Runfile Installer to uninstall a previous ROCm or AMDGPU driver installation.

- uninstall-rocm (target=<directory>)

  This option configures the ROCm Runfile Installer to uninstall a previous ROCm installation.

  The parameter target=<directory> is optional for the uninstall-rocm option. If it's set, the uninstall looks for a pre-existing ROCm installation at the specified directory path and attempts to remove it.

  To uninstall ROCm from the default location ($PWD/rocm), use the following command line:

```
bash rocm-installer.run uninstall-rocm
```

> **Note**
>
> This command matches the case where ROCm was installed without the target=<directory> option. To uninstall ROCm from a specific location, append target=<directory>. For example, if ROCm was previously installed to /home/amd/myrocm, use the following command line:
>
> ```
> bash rocm-installer.run uninstall-rocm target="/home/amd/myrocm"
> ```

> **Note**
>
> The uninstall-rocm option can only remove ROCm if it was installed using the ROCm Runfile Installer. Traditional package-based ROCm installs will not be removed.

- uninstall-amdgpu

  This option configures the ROCm Runfile Installer to uninstall a previous AMDGPU driver installation.

  To uninstall the AMDGPU driver from the system, use the following command line:

  ```
  bash rocm-installer.run uninstall-amdgpu
  ```

  > **Note**
  >
  > The uninstall-amdgpu option can only remove the AMDGPU driver if it was installed using the ROCm Runfile Installer. Traditional package-based AMDGPU driver installs will not be removed.

2.2.6.7.1.6 Information and debug options

The ROCm Runfile Installer command line interface includes options for information output or debugging.

- findrocm

  This information option searches the install system for any existing installations of ROCm. Any install locations are output to the terminal.

  Use findrocm as a standalone <options> parameter:

  ```
  bash rocm-installer.run findrocm
  ```

- complist

  This information option lists the version of each ROCm component included in the installer. The complist option causes the Runfile Installer to quit after listing the ROCm components.

  Use complist as a standalone <options> parameter:

  ```
  bash rocm-installer.run complist
  ```

- prompt

  This debug option enables user prompts in the installer. At specific, critical points of the installation process, the installer halts execution and prompts the user to either continue with the install or exit.

- verbose

  This debug option enables verbose logging during ROCm Runfile Installer execution.

For example, to install ROCm with user prompts and verbose logging, the command line is as follows:

```
bash rocm-installer.run target="/" rocm prompt verbose
```

2.2.6.8 Log files

By default, the ROCm Runfile Installer GUI and command line interfaces record the output execution in log files. These log files are created in the rocm-installer/logs directory.

The rocm-installer directory is created when the ROCm Runfile Installer self-extracts to the current working directory where it is being executed.

2.3 Post-installation instructions

After installing ROCm, follow these steps to finalize and validate the installation.

2.3.1 Environment Configuration

2.3.1.1 1. Configure ROCm shared objects

Configure the system linker by specifying where to find the shared objects (.so files) for ROCm applications.

```
sudo tee --append /etc/ld.so.conf.d/rocm.conf <<EOF
/opt/rocm/lib
/opt/rocm/lib64
EOF
sudo ldconfig
```

2.3.1.2 2. Configure ROCm PATH

Configure the path to the ROCm binary using one of the following Linux utilities or manually update the PATH variable. The ROCm installation process adds the ROCm executables to these systems, provided they are installed on the system.

Option A: update-alternatives

The update-alternatives utility is available on most Linux distributions. It helps manage multiple versions of a command or program. For more information about update-alternatives, see Linux man.

To use update-alternatives, follow these steps:

1. Display a list of all ROCm versions available:

   ```
   sudo update-alternatives --display rocm
   ```

2. If multiple ROCm versions are installed, switch between them using this command and selecting the ROCm version:

   ```
   sudo update-alternatives --config rocm
   ```

Option B: environment-modules

The environment-modules tool simplifies shell initialization. It lets you modify your session environment using module files. For more information, see Environment Modules.

> **Note**
>
> The environment-modules package should be installed on the system before ROCm can be configured using modules. For more information, see prerequisites.

To use environment-modules, follow these instructions:

1. Enable environment-modules:

```
source /etc/profile.d/modules.sh
```

2. Display a list of all available modules (including ROCm modules):

```
module avail
```

3. Display a list of all currently loaded modules:

```
module list
```

4. If multiple ROCm versions are installed, and no other ROCm modules are currently loaded, set ROCm by version:

```
module load rocm/6.4.1
```

5. If multiple ROCm versions are installed with a current ROCm module in use, switch to another ROCm version as follows:

```
module switch rocm/6.4.1
```

> **Note**
>
> If modules are used for ROCm, any update-alternatives ROCm setting will be overwritten for the terminal session.

Option C: PATH

If update-alternatives or environment-modules are not available on the system, configure the ROCm path by setting the PATH variable to /opt/rocm-<version>/bin.

```
export PATH=$PATH:/opt/rocm-6.4.1/bin
```

2.3.1.3 3. Configure LD_LIBRARY_PATH

> **Important**
>
> This step is required for version-specific or multi-version installations.

```
export LD_LIBRARY_PATH=/opt/rocm-6.4.1/lib
```

2.3.2 Install verification

Once ROCm has been configured, validate the installation.

2.3.2.1 1. Verify the package installation

Use the package manager to validate the list of ROCm component packages installed on the system. If package installation was successful, the list will contain rocm* and hip* packages currently installed on the system.

Ubuntu

```
apt list --installed
```

Debian

```
apt list --installed
```

RHEL

```
dnf list installed
```

OL

```
dnf list installed
```

SLES

```
zypper search --installed-only
```

AZL

```
tdnf list installed
```

2.3.2.2 2. Verify the ROCm installation

Use the following ROCm tools to verify that installation was successful:

```
rocminfo
clinfo
```

Both rocminfo and clinfo should output attributes for the ROCm system configuration if installation was successful. For additional testing of ROCm functionality, try rocm-examples.

2.3.3 Troubleshooting

1. What if environment-modules is not installed before installing ROCm?

   You can still install environment-modules package after installing ROCm. However, no ROCm modules will be listed for the module avail command. As an alternative to the standard method of loading the ROCm modules, you can load each version-specific module directly from the /opt/rocm-<version> directory:

```
module load /opt/rocm-6.4.1/lib/rocmmod
```

2. Will the ROCm path configuration persist once I set it?

   - If you are using update-alternatives to configure ROCm, then yes, the currently set configuration will persist even after a system reboot.
   - If you are using environment-modules to configure ROCm, then no, the current set configuration will only last for the current terminal session.

CHAPTER

THREE

PYTORCH ON ROCM

PyTorch is an open-source tensor library designed for deep learning. PyTorch on ROCm provides mixed-precision and large-scale training using our MIOpen and RCCL libraries.

To install PyTorch for ROCm, you have the following options:

- Using a Docker image with PyTorch pre-installed (recommended)
  - Docker image support
- Using a wheels package
- Using the PyTorch ROCm base Docker image
- Using the PyTorch upstream Dockerfile

For hardware, software, and third-party framework compatibility between ROCm and PyTorch, see the following resources:

- System requirements (Linux)
- PyTorch compatibility

3.1 Using a Docker image with PyTorch pre-installed

To install ROCm on bare metal, follow ROCm installation overview. The recommended option to get a PyTorch environment is through Docker.

Using Docker provides portability and access to a prebuilt Docker image that has been rigorously tested within AMD. This can also save compilation time and should perform as tested and mitigate potential installation issues. See Docker image support

1. Download the latest public PyTorch Docker image.

   ```
   docker pull rocm/pytorch:latest
   ```

   Important

   The rocm/pytorch:latest and rocm/pytorch:latest-release tags point to a Docker image with the latest ROCm-tested release of PyTorch.

   The rocm/pytorch:latest-release-preview tag points to a more recent PyTorch version with limited testing on ROCm.

   You can download Docker images with specific ROCm, PyTorch, and operating system versions. See the available tags on Docker Hub.

2. Start a Docker container using the image.

```
docker run -it --cap-add=SYS_PTRACE --security-opt seccomp=unconfined \
--device=/dev/kfd --device=/dev/dri --group-add video \
--ipc=host --shm-size 8G rocm/pytorch:latest
```

> **Note**
>
> This will automatically download the image if it does not exist on the host. You can also pass the -v argument to mount any data directories from the host onto the container.

3.1.1 Docker image support

AMD validates and publishes ready-made PyTorch images with ROCm backends on Docker Hub. The following Docker image tags and associated inventories are validated for ROCm 6.4.1.

PyTorch 2.6.0

Ubuntu 24.04

Tag
: rocm/pytorch:rocm6.4.1_ubuntu24.04_py3.12_pytorch_release_2.6.0

> **Note**
>
> As of ROCm 6.2.1, rocm/pytorch:latest points to a Docker image with the latest ROCm tested release version of PyTorch (for example, version 2.4), similar to rocm/pytorch:latest-release tag. See Using a Docker image with PyTorch pre-installed for more information.

Inventory

- ROCm 6.4.1
- Python 3.12
- PyTorch 2.6.0
- Apex 1.6.0
- torchvision 0.21.0
- TensorBoard 2.13.0
- MAGMA
- UCX 1.16.0
- OMPI 4.1.6-7ubuntu2
- OFED

Ubuntu 22.04

Tag
: rocm/pytorch:rocm6.4.1_ubuntu22.04_py3.10_pytorch_release_2.6.0

Inventory

- ROCm 6.4.1

- Python 3.10
- PyTorch 2.6.0
- Apex 1.6.0
- torchvision 0.21.0
- TensorBoard 2.13.0
- MAGMA
- UCX 1.12.1~rc2-1
- OMPI 4.1.2-2ubuntu1
- OFED

PyTorch 2.5.1

Ubuntu 24.04

Tag
rocm/pytorch:rocm6.4.1_ubuntu24.04_py3.12_pytorch_release_2.5.1

Inventory

- ROCm 6.4.1
- Python 3.12
- PyTorch 2.5.1
- Apex 1.5.0
- torchvision 0.20.1
- TensorBoard 2.13.0
- MAGMA
- UCX 1.16.0+ds-5ubuntu1
- OMPI 4.1.6-7ubuntu2
- OFED

Ubuntu 22.04

Tag
rocm/pytorch:rocm6.4.1_ubuntu24.04_py3.10_pytorch_release_2.5.1

Inventory

- ROCm 6.4.1
- Python 3.10
- PyTorch 2.5.1
- Apex 1.5.0
- torchvision 0.20.1
- TensorBoard 2.13.0
- MAGMA
- UCX 1.12.1~rc2-1

- OMPI 4.1.2-2ubuntu1
- OFED

PyTorch 2.4.1

Ubuntu 24.04

Tag
rocm/pytorch:rocm6.4.1_ubuntu24.04_py3.12_pytorch_release_2.4.1

Inventory

- ROCm 6.4.1
- Python 3.12
- PyTorch 2.4.1
- Apex 1.4.0
- torchvision 0.19.0
- TensorBoard 2.13.0
- MAGMA
- UCX 1.16.0+ds-5ubuntu1
- OMPI 4.1.6-7ubuntu2
- OFED

Ubuntu 22.04

Tag
rocm/pytorch:rocm6.4.1_ubuntu22.04_py3.10_pytorch_release_2.4.1

Inventory

- ROCm 6.4.1
- Python 3.10
- PyTorch 2.4.1
- Apex 1.4.0
- torchvision 0.19.0
- TensorBoard 2.13.0
- MAGMA
- UCX 1.12.1~rc2-1
- OMPI 4.1.2-2ubuntu1
- OFED

PyTorch 2.3.0

Ubuntu 24.04

Tag
rocm/pytorch:rocm6.4.1_ubuntu24.04_py3.12_pytorch_release_2.3.0

Inventory

- ROCm 6.4.1
- Python 3.12
- PyTorch 2.3.0
- Apex 1.3.0
- torchvision 0.18.0
- TensorBoard 2.13.0
- MAGMA
- UCX 1.16.0+ds-5ubuntu1
- OMPI 4.1.6-7ubuntu2
- OFED

Ubuntu 22.04

Tag
: rocm/pytorch:rocm6.4.1_ubuntu22.04_py3.10_pytorch_release_2.3.0

Inventory

- ROCm 6.4.1
- Python 3.10
- PyTorch 2.3.0
- Apex 1.3.0
- torchvision 0.18.0
- TensorBoard 2.13.0
- MAGMA
- UCX 1.12.1~rc2-1
- OMPI 4.1.2-2ubuntu1
- OFED

3.2 Using a wheels package

PyTorch supports the ROCm platform by providing tested wheels packages. To access this feature, go to pytorch.org/get-started/locally/. For the correct wheels command, you must select Linux, Python, pip, and ROCm in the matrix.

> **Note**
>
> The available ROCm release varies between the PyTorch Build of Stable or Nightly. More recent releases are generally available through the Nightly builds.

1. Choose one of the following three options:

   Option 1:

a. Download a base Docker image with the correct ROCm version.

Base OS	Docker Image
Ubuntu 22.04	rocm/dev-ubuntu-22.04
Ubuntu 24.04	rocm/dev-ubuntu-24.04

b. Pull the selected image.

```
docker pull rocm/dev-ubuntu-22.04:latest
```

c. Start a Docker container using the downloaded image.

```
docker run -it --device=/dev/kfd --device=/dev/dri --group-add video rocm/dev-ubuntu-22.
↪04:latest
```

Option 2:

a. Select a base OS Docker image. Check System requirements (Linux).

b. Pull selected base OS image (Ubuntu 22.04, for example).

```
docker pull ubuntu:22.04
```

c. Start a Docker container using the downloaded image.

```
docker run -it --device=/dev/kfd --device=/dev/dri --group-add video ubuntu:22.04
```

d. Install ROCm using the directions in the ROCm installation overview section.

Option 3:

Install on bare metal. Check System requirements (Linux) and install ROCm using the directions in the ROCm installation overview section.

2. Install the required dependencies for the wheels package.

```
sudo apt update
sudo apt install libjpeg-dev python3-dev python3-pip
pip3 install wheel setuptools
```

3. Install torch, torchvision, and torchaudio, as specified in the installation matrix.

> **Note**
>
> The following command uses the ROCm 6.4.0 PyTorch wheel. If you want a different version of ROCm, modify the command accordingly.

```
pip3 install --pre torch torchvision torchaudio --index-url https://download.pytorch.org/whl/
↪nightly/rocm6.4/
```

4. (Optional) Use MIOpen kdb files with ROCm PyTorch wheels.

   PyTorch uses MIOpen for machine learning primitives, which are compiled into kernels at runtime. Runtime compilation causes a small warm-up phase when starting PyTorch, and MIOpen kdb files contain precompiled kernels that can speed up application warm-up phases.

MIOpen kdb files can be used with ROCm PyTorch wheels. However, the kdb files need to be placed in a specific location with respect to the PyTorch installation path. A helper script simplifies this task by taking the ROCm version and GPU architecture as inputs. This works for Ubuntu.

You can download the helper script here: install_kdb_files_for_pytorch_wheels.sh, or use:

```
wget https://raw.githubusercontent.com/wiki/ROCm/pytorch/files/install_kdb_files_for_pytorch_
↪wheels.sh
```

After installing ROCm PyTorch wheels, run the following code:

```
#Optional: replace 'gfx90a' with your architecture and 6.2.4 with your preferred ROCm version
export GFX_ARCH=gfx90a

#Optional
export ROCM_VERSION=6.2.4

./install_kdb_files_for_pytorch_wheels.sh
```

3.3 Using the PyTorch ROCm base Docker image

The pre-built base Docker image has all dependencies installed, including:

- ROCm
- torchvision
- Conda packages
- The compiler toolchain

Additionally, a particular environment flag (BUILD_ENVIRONMENT) is set, which is used by the build scripts to determine the configuration of the build environment.

1. Download the Docker image. This is the base image, which does not contain PyTorch.

   ```
   docker pull rocm/pytorch:latest-base
   ```

2. Start a Docker container using the downloaded image.

   ```
   docker run -it --cap-add=SYS_PTRACE --security-opt seccomp=unconfined --device=/dev/kfd --
   ↪device=/dev/dri --group-add video --ipc=host --shm-size 8G rocm/pytorch:latest-base
   ```

   You can also pass the -v argument to mount any data directories from the host onto the container.

Inside the docker container, run the following steps:

3. Clone the PyTorch repository.

   ```
   cd ~
   git clone https://github.com/pytorch/pytorch.git
   cd pytorch
   git submodule update --init --recursive
   ```

4. Set ROCm architecture (optional).

> **Note**
>
> By default in the rocm/pytorch:latest-base image, PyTorch builds simultaneously for the following architectures:
>
> - gfx900
> - gfx906
> - gfx908
> - gfx90a
> - gfx1030
> - gfx1100
> - gfx1101
> - gfx940
> - gfx941
> - gfx942

If you want to compile only for your microarchitecture (uarch), run:

```
export PYTORCH_ROCM_ARCH=<uarch>
```

Where <uarch> is the architecture reported by the rocminfo command.

To find your uarch, run:

```
rocminfo | grep gfx
```

5. Build PyTorch.

```
.ci/pytorch/build.sh
```

This converts PyTorch sources for HIP compatibility and builds the PyTorch framework.

To check if your build is successful, run:

```
echo $? # should return 0 if success
```

3.4 Using the PyTorch upstream Dockerfile

If you don't want to use a prebuilt base Docker image, you can build a custom base Docker image using scripts from the PyTorch repository. This uses a standard Docker image from operating system maintainers and installs all the required dependencies, including:

- ROCm
- torchvision
- Conda packages
- The compiler toolchain

1. Clone the PyTorch repository.

```
cd ~
git clone https://github.com/pytorch/pytorch.git
cd pytorch
git submodule update --init --recursive
```

2. Build the PyTorch Docker image.

```
cd .ci/docker
./build.sh pytorch-linux-<os-version>-rocm<rocm-version>-py<python-version> -t rocm/
↪pytorch:build_from_dockerfile
```

Where:

- <os-version> = ubuntu20.04 (or focal), ubuntu22.04 (or jammy)
- <rocm-version> = 6.0, 6.1, 6.2
- <python-version> = 3.8 - 3.11

To verify that your image was successfully created, run:

```
docker image ls rocm/pytorch:build_from_dockerfile
```

If successful, the output looks like this:

```
REPOSITORY    TAG                    IMAGE ID       CREATED          SIZE
rocm/pytorch  build_from_dockerfile   17071499be47   2 minutes ago    32.8GB
```

3. Start a Docker container using the image with the mounted PyTorch folder.

```
docker run -it --cap-add=SYS_PTRACE --security-opt seccomp=unconfined \
--user root --device=/dev/kfd --device=/dev/dri \
--group-add video --ipc=host --shm-size 8G \
-v ~/pytorch:/pytorch rocm/pytorch:build_from_dockerfile
```

You can also pass the -v argument to mount any data directories from the host onto the container.

4. Go to the PyTorch directory.

```
cd /pytorch
```

5. Set ROCm architecture.

To determine your AMD architecture, run:

```
rocminfo | grep gfx
```

The result looks like this (for gfx1030 architecture):

```
Name:                    gfx1030
Name:                    amdgcn-amd-amdhsa--gfx1030
```

Set the PYTORCH_ROCM_ARCH environment variable to specify the architectures you want to build PyTorch for.

```
export PYTORCH_ROCM_ARCH=<uarch>
```

where <uarch> is the architecture reported by the rocminfo command.

6. Build PyTorch.

```
.ci/pytorch/build.sh
```

This converts PyTorch CUDA sources to HIP and builds the PyTorch framework.

To check if your build is successful, run:

```
echo $? # should return 0 if success
```

3.5 Testing the PyTorch installation

You can use PyTorch unit tests to validate your PyTorch installation. If you used a prebuilt PyTorch Docker image from AMD ROCm Docker Hub or installed an official wheels package, validation tests are not necessary.

If you want to manually run unit tests to validate your PyTorch installation fully, follow these steps:

1. Import the torch package in Python to test if PyTorch is installed and accessible.

> **Note**
>
> Do not run the following command from the PyTorch home directory.

```
python3 -c 'import torch' 2> /dev/null && echo 'Success' || echo 'Failure'
```

2. Check if the GPU is accessible from PyTorch. In the PyTorch framework, torch.cuda is a generic way to access the GPU. This can only access an AMD GPU if one is available.

```
python3 -c 'import torch; print(torch.cuda.is_available())'
```

3. Run unit tests to validate the PyTorch installation fully.

> **Note**
>
> You must run the following command from the PyTorch home directory.

```
PYTORCH_TEST_WITH_ROCM=1 python3 test/run_test.py --verbose \
--include test_nn test_torch test_cuda test_ops \
test_unary_ufuncs test_binary_ufuncs test_autograd
```

This command ensures that the required environment variable is set to skip certain unit tests for ROCm. This also applies to wheel installs in a non-controlled environment.

> **Note**
>
> Make sure your PyTorch source code corresponds to the PyTorch wheel or the installation in the Docker image. Incompatible PyTorch source code can give errors when running unit tests.

Some tests may be skipped, as appropriate, based on your system configuration. ROCm doesn't support all PyTorch features; tests that evaluate unsupported features are skipped. Other tests might be skipped, depending on the host or GPU memory and the number of available GPUs.

If the compilation and installation are correct, all tests will pass.

4. (Optional) Run individual unit tests.

```
PYTORCH_TEST_WITH_ROCM=1 python3 test/test_nn.py --verbose
```

You can replace test_nn.py with any other test set.

3.6 Running a basic PyTorch example

The PyTorch examples repository provides basic examples that exercise the functionality of your framework.

Two of our favorite testing databases are:

- MNIST (Modified National Institute of Standards and Technology): A database of handwritten digits that can be used to train a Convolutional Neural Network for handwriting recognition.
- ImageNet: A database of images that can be used to train a network for visual object recognition.

3.6.1 MNIST PyTorch example

1. Clone the PyTorch examples repository.

```
git clone https://github.com/pytorch/examples.git
```

2. Go to the MNIST example folder.

```
cd examples/mnist
```

3. Follow the instructions in the README.md file in this folder to install the requirements. Then run:

```
python3 main.py
```

This generates the following output:

```
...
Train Epoch: 14 [58240/60000 (97%)]     Loss: 0.010128
Train Epoch: 14 [58880/60000 (98%)]     Loss: 0.001348
Train Epoch: 14 [59520/60000 (99%)]     Loss: 0.005261

Test set: Average loss: 0.0252, Accuracy: 9921/10000 (99%)
```

3.6.2 ImageNet PyTorch example

1. Clone the PyTorch examples repository (if you didn't already do this in the preceding MNIST example).

```
git clone https://github.com/pytorch/examples.git
```

2. Go to the ImageNet example folder.

```
cd examples/imagenet
```

3. Follow the instructions in the README.md file in this folder to install the Requirements. Then run:

```
python3 main.py
```

3.7 Troubleshooting

- What to do if you get the following error when trying to run PyTorch:

  ```
  hipErrorNoBinaryForGPU: Unable to find code object for all current devices!
  ```

  The error denotes that the installation of PyTorch and/or other dependencies or libraries do not support the current GPU. To workaround this issue, use the following steps:

  1. Confirm that the hardware supports the ROCm stack. Refer to System requirements (Linux) and System requirements for Windows.
  2. Determine the gfx target.

     ```
     rocminfo | grep gfx
     ```

  3. Check if PyTorch is compiled with the correct gfx target.

     ```
     TORCHDIR=$( dirname $( python3 -c 'import torch; print(torch.__file__)' ) )
     roc-obj-ls -v $TORCHDIR/lib/libtorch_hip.so # check for gfx target
     ```

     > **Note**
     >
     > Recompile PyTorch with the right gfx target if compiling from the source if the hardware is not supported.

- What if you are unable to access Docker or GPU in user accounts?

  Ensure that the user is added to docker, video, and render Linux groups as described in Configuring permissions for GPU access.

- Can you install PyTorch directly on bare metal?

  Bare-metal installation of PyTorch is supported through wheels. For more information, see Using a wheels package.

- How do you profile PyTorch workloads?

  Use the PyTorch Profiler as described in PyTorch Profiler to profile GPU kernels on ROCm.

CHAPTER

FOUR

TENSORFLOW ON ROCM

TensorFlow is an open-source library for solving machine learning, deep learning, and AI problems. It can solve many problems across different sectors and industries, but primarily focuses on neural network training and inference. It is one of the most popular and in-demand frameworks and is very active in open-source contribution and development.

To install TensorFlow for ROCm, you have the following options:

- Using a Docker image with TensorFlow pre-installed (recommended)
  - Docker image support
- Using a wheels package

For hardware, software, and third-party framework compatibility between ROCm and TensorFlow, see the following resources:

- System requirements (Linux)
- TensorFlow compatibility

> **Note**
>
> As of ROCm 6.1, tensorflow-rocm packages are found at https://repo.radeon.com/rocm/manylinux. Prior to ROCm 6.1, packages were found at https://pypi.org/project/tensorflow-rocm.

ROCm version	TensorFlow version
6.4.x	2.18.1, 2.17.1, 2.16.2
6.3.x	2.17.0, 2.16.2 2.15.1
6.2.x	2.16.1, 2.15.1, 2.14.1
6.1.x	2.15.0, 2.14.0, 2.13.1
6.0.x	2.14.0, 2.13.1 2.12.1

4.1 Using a Docker image with TensorFlow pre-installed

To install ROCm on bare metal, follow ROCm installation overview. The recommended option to get a TensorFlow environment is through Docker.

Using Docker provides portability and access to a prebuilt Docker image that has been rigorously tested within AMD. This can also save compilation time and should perform as tested and mitigate potential installation issues. See Docker image support

Follow these steps:

1. Pull the latest public TensorFlow Docker image.

```
docker pull rocm/tensorflow:latest
```

2. Once you have pulled the image, run it by using the command below:

```
docker run -it --network=host --device=/dev/kfd --device=/dev/dri \
--ipc=host --shm-size 16G --group-add video --cap-add=SYS_PTRACE \
--security-opt seccomp=unconfined rocm/tensorflow:latest
```

4.1.1 Docker image support

AMD validates and publishes ready-made TensorFlow images with ROCm backends on Docker Hub. The following Docker image tags and associated inventories are validated for ROCm 6.4.1.

TensorFlow 2.18.1

Ubuntu 24.04

Tag
rocm/tensorflow:rocm6.4.1-py3.12-tf2.18-dev

Inventory

- ROCm 6.4.1
- Python 3.12
- tensorflow-rocm 2.18.1
- TensorBoard 2.18.0

Tag
rocm/tensorflow:rocm6.4.1-py3.12-tf2.18-runtime

Inventory

- ROCm 6.4.1
- Python 3.12
- tensorflow-rocm 2.18.1
- TensorBoard 2.18.0

Ubuntu 22.04

Tag
rocm/tensorflow:rocm6.4.1-py3.10-tf2.18-dev

Inventory

- ROCm 6.4.1
- Python 3.10
- tensorflow-rocm 2.18.1
- TensorBoard 2.18.0

Tag
rocm/tensorflow:rocm6.4.1-py3.10-tf2.18-runtime

Inventory

- ROCm 6.4.1
- Python 3.10
- tensorflow-rocm 2.18.1
- TensorBoard 2.18.0

TensorFlow 2.17.1

Ubuntu 24.04

Tag
rocm/tensorflow:rocm6.4.1-py3.12-tf2.17-dev

Inventory

- ROCm 6.4.1
- Python 3.12
- tensorflow-rocm 2.17.1
- TensorBoard 2.17.1

Tag
rocm/tensorflow:rocm6.4.1-py3.12-tf2.17-runtime

Inventory

- ROCm 6.4.1
- Python 3.12
- tensorflow-rocm 2.17.1
- TensorBoard 2.17.1

Ubuntu 22.04

Tag
rocm/tensorflow:rocm6.4.1-py3.10-tf2.17-dev

Inventory

- ROCm 6.4.1
- Python 3.10
- tensorflow-rocm 2.17.1
- TensorBoard 2.17.1

Tag
rocm/tensorflow:rocm6.4.1-py3.10-tf2.17-runtime

Inventory

- ROCm 6.4.1
- Python 3.10
- tensorflow-rocm 2.17.1
- TensorBoard 2.17.1

TensorFlow 2.16.2

Ubuntu 24.04

Tag
rocm/tensorflow:rocm6.4.1-py3.12-tf2.16-dev

Inventory

- ROCm 6.4.1
- Python 3.12
- tensorflow-rocm 2.16.2
- TensorBoard 2.16.2

Tag
rocm/tensorflow:rocm6.4.1-py3.12-tf2.16-runtime

Inventory

- ROCm 6.4.1
- Python 3.12
- tensorflow-rocm 2.16.2
- TensorBoard 2.16.2

Ubuntu 22.04

Tag
rocm/tensorflow:rocm6.4.1-py3.10-tf2.16-dev

Inventory

- ROCm 6.4.1
- Python 3.10
- tensorflow-rocm 2.16.2
- TensorBoard 2.16.2

Tag
rocm/tensorflow:rocm6.4.1-py3.10-tf2.16-runtime

Inventory

- ROCm 6.4.1
- Python 3.10
- tensorflow-rocm 2.16.2
- TensorBoard 2.16.2

4.2 Using a wheels package

To install TensorFlow using the wheels package, use the following command.

```
pip install --user tensorflow-rocm==[wheel-version] -f [repo] --upgrade
```

- [wheel-version] is the TensorFlow version.

- [repo] is https://repo.radeon.com/rocm/manylinux/rocm-rel-X.Y/ for versions 6.1 and later, where X.Y indicates the ROCm version.

> **Note**
>
> Prior to ROCm 6.1, [wheel-version] followed the <TensorFlowVersion>.<ROCmVersion> format.

4.3 Testing the TensorFlow installation

To test the installation of TensorFlow, run the container as specified in Installing TensorFlow. Ensure you have access to the Python shell in the Docker container.

```
python -c 'import tensorflow' 2> /dev/null && echo 'Success' || echo 'Failure'
```

4.4 Running a basic TensorFlow example

To quickly validate your TensorFlow environment, run a basic TensorFlow example.

The MNIST dataset is a collection of handwritten digits that may be used to train a Convolutional Neural Network (CNN) for handwriting recognition. This dataset is included with your TensorFlow installation.

Run the following sample code to load the MNIST dataset, then train and evaluate it.

```
import tensorflow as tf
print("TensorFlow version:", tf.__version__)
mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0
model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10)
])
predictions = model(x_train[:1]).numpy()
tf.nn.softmax(predictions).numpy()
loss_fn = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
loss_fn(y_train[:1], predictions).numpy()
model.compile(optimizer='adam',
              loss=loss_fn,
              metrics=['accuracy'])
model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test,  y_test, verbose=2)
```

If successful, you should see the following output indicating the image classifier is now trained to around 98% accuracy on this dataset.

```
I0000 00:00:1716848656.190857     212 device_compiler.h:186] Compiled cluster using XLA!  This line is logged at
fetime of the process.
1858/1875 [============================>.] - ETA: 0s - loss: 0.2965 - accuracy: 0.91442024-05-27 22:24:18.503331
ommon_runtime/gpu_fusion_pass.cc:508] ROCm Fusion is enabled.
1875/1875 [==============================] - 3s 924us/step - loss: 0.2952 - accuracy: 0.9148
2024-05-27 22:24:18.504658: I tensorflow/core/common_runtime/gpu_fusion_pass.cc:508] ROCm Fusion is enabled.
Epoch 2/5
1875/1875 [==============================] - 2s 923us/step - loss: 0.1439 - accuracy: 0.9571
Epoch 3/5
1875/1875 [==============================] - 2s 924us/step - loss: 0.1098 - accuracy: 0.9668
Epoch 4/5
1875/1875 [==============================] - 2s 924us/step - loss: 0.0900 - accuracy: 0.9724
Epoch 5/5
1875/1875 [==============================] - 2s 925us/step - loss: 0.0762 - accuracy: 0.9760
2024-05-27 22:24:25.470030: I tensorflow/core/common_runtime/gpu_fusion_pass.cc:508] ROCm Fusion is enabled.
2024-05-27 22:24:25.470639: I tensorflow/core/common_runtime/gpu_fusion_pass.cc:508] ROCm Fusion is enabled.
2024-05-27 22:24:25.473927: I tensorflow/core/common_runtime/gpu_fusion_pass.cc:508] ROCm Fusion is enabled.
2024-05-27 22:24:25.474474: I tensorflow/core/common_runtime/gpu_fusion_pass.cc:508] ROCm Fusion is enabled.
2024-05-27 22:24:25.479210: I tensorflow/core/common_runtime/gpu_fusion_pass.cc:508] ROCm Fusion is enabled.
2024-05-27 22:24:25.479879: I tensorflow/core/common_runtime/gpu_fusion_pass.cc:508] ROCm Fusion is enabled.
2024-05-27 22:24:25.487077: I tensorflow/core/common_runtime/gpu_fusion_pass.cc:508] ROCm Fusion is enabled.
2024-05-27 22:24:25.487564: I tensorflow/core/common_runtime/gpu_fusion_pass.cc:508] ROCm Fusion is enabled.
2024-05-27 22:24:25.490836: I tensorflow/core/common_runtime/gpu_fusion_pass.cc:508] ROCm Fusion is enabled.
2024-05-27 22:24:25.491442: I tensorflow/core/common_runtime/gpu_fusion_pass.cc:508] ROCm Fusion is enabled.
2024-05-27 22:24:25.491931: I tensorflow/core/common_runtime/gpu_fusion_pass.cc:508] ROCm Fusion is enabled.
2024-05-27 22:24:25.492460: I tensorflow/core/common_runtime/gpu_fusion_pass.cc:508] ROCm Fusion is enabled.
2024-05-27 22:24:25.494832: I tensorflow/core/common_runtime/gpu_fusion_pass.cc:508] ROCm Fusion is enabled.
2024-05-27 22:24:25.497105: I tensorflow/core/common_runtime/gpu_fusion_pass.cc:508] ROCm Fusion is enabled.
2024-05-27 22:24:25.499508: I tensorflow/core/common_runtime/gpu_fusion_pass.cc:508] ROCm Fusion is enabled.
2024-05-27 22:24:25.501408: I tensorflow/core/common_runtime/gpu_fusion_pass.cc:508] ROCm Fusion is enabled.
2024-05-27 22:24:25.613366: I tensorflow/core/common_runtime/gpu_fusion_pass.cc:508] ROCm Fusion is enabled.
```

CHAPTER

FIVE

JAX ON ROCM

This directory provides setup instructions and necessary files to build, test, and run JAX with ROCm support in a Docker environment, suitable for both runtime and CI workflows. Explore the following methods to use or build JAX on ROCm.

For hardware, software, and third-party framework compatibility between ROCm and JAX, see the following resources:

- System requirements (Linux)
- JAX compatibility

5.1 Using a prebuilt Docker image

The ROCm JAX team provides prebuilt Docker images, which is the simplest way to use JAX on ROCm. These images are available on Docker Hub and come with JAX configured for ROCm.

1. To pull the latest ROCm JAX Docker image, run:

```
docker pull rocm/jax-community:latest
```

> **Note**
>
> For specific versions of JAX, review the periodically pushed Docker images at ROCm JAX Community on Docker Hub.
>
> Additional Docker images are available at ROCm JAX on Docker Hub. These contain the latest ROCm version but might use an older version of JAX.

2. Once the image is downloaded, launch a container using the following command:

```
docker run -it -d --network=host --device=/dev/kfd --device=/dev/dri --ipc=host --shm-size 64G \
--group-add video --cap-add=SYS_PTRACE --security-opt seccomp=unconfined -v $(pwd):/jax_
↪dir \
--name rocm_jax rocm/jax-community:latest /bin/bash

docker attach rocm_jax
```

> **Tip**

- The --shm-size parameter allocates shared memory for the container. Adjust it based on your system's resources if needed.
- Replace $(pwd) with the absolute path to the directory you want to mount inside the container.
- If you prefer to use rocm/jax, remember to replace rocm/jax-community with rocm/jax.

3. Verify the installation of ROCm JAX. See Testing your JAX installation with ROCm.

5.2 Using a ROCm base Docker image and installing JAX

If you prefer to use the ROCm Ubuntu image or already have a ROCm Ubuntu container, follow these steps to install JAX in the container.

1. Pull the ROCm Ubuntu Docker image. For example, use the following command to pull the ROCm Ubuntu image:

```
docker pull rocm/dev-ubuntu-22.04:6.3-complete
```

2. Launch the Docker container. After pulling the image, launch a container using this command:

```
docker run -it -d --network=host --device=/dev/kfd --device=/dev/dri --ipc=host --shm-size 64G \
--group-add video --cap-add=SYS_PTRACE --security-opt seccomp=unconfined -v $(pwd):/jax_
↪dir \
--name rocm_jax rocm/dev-ubuntu-22.04:6.3-complete /bin/bash
docker attach rocm_jax
```

3. Install the latest version of JAX. Inside the running container, install the required version of JAX with ROCm support using pip:

```
pip3 install jax[rocm]
```

4. Verify the installed JAX version. Check whether the correct version of JAX and its ROCm plugins are installed.

```
pip3 freeze | grep jax
```

Expected output:

```
jax==0.4.35
jax-rocm60-pjrt==0.4.35
jax-rocm60-plugin==0.4.35
jaxlib==0.4.35
```

5. Explicitly set the LLVM_PATH environment variable. This helps XLA find ld.lld in the PATH at runtime.

```
export LLVM_PATH=/opt/rocm/llvm
```

6. Verify the installation of ROCm JAX. See Testing your JAX installation with ROCm.

5.3 Install JAX on bare-metal or a custom container

Follow these steps if you prefer to install ROCm manually on your host system or in a custom container.

1. Install ROCm. Follow the ROCm installation guide to install ROCm on your system.

   Once installed, verify your ROCm installation using:

```
rocm-smi
```

```
========================================= ROCm System Management Interface =========================================
====================================================================================================
Concise Info
====================================================================================================
Device  [Model : Revision]    Temp        Power     Partitions      SCLK     MCLK     Fan  Perf  PwrCap  VRAM%  GPU%
        Name (20 chars)       (Junction)  (Socket)  (Mem, Compute)
====================================================================================================
0       [0x74a1 : 0x00]       50.0°C      170.0W    NPS1, SPX       131Mhz   900Mhz   0%   auto  750.0W    0%    0%
        AMD Instinct MI300X
1       [0x74a1 : 0x00]       51.0°C      176.0W    NPS1, SPX       132Mhz   900Mhz   0%   auto  750.0W    0%    0%
        AMD Instinct MI300X
2       [0x74a1 : 0x00]       50.0°C      177.0W    NPS1, SPX       132Mhz   900Mhz   0%   auto  750.0W    0%    0%
        AMD Instinct MI300X
3       [0x74a1 : 0x00]       53.0°C      176.0W    NPS1, SPX       132Mhz   900Mhz   0%   auto  750.0W    0%    0%
        AMD Instinct MI300X
====================================================================================================
========================================= End of ROCm SMI Log ==========================================
```

2. Install the required version of JAX with ROCm support using pip:

```
pip3 install jax[rocm]
```

3. Verify the installed JAX version. Check whether the correct version of JAX and its ROCm plugins are installed.

```
pip3 freeze | grep jax
```

4. Explicitly set the LLVM_PATH environment variable.

```
export LLVM_PATH=/opt/rocm/llvm
```

5. Verify the installation of ROCm JAX.

   Run the following commands to verify that ROCm JAX is installed correctly:

```
python3 -c "import jax; print(jax.devices())"
python3 -c "import jax.numpy as jnp; x = jnp.arange(5); print(x)"
```

Expected output:

```
[RocmDevice(id=0), RocmDevice(id=1), RocmDevice(id=2), RocmDevice(id=3)]
```

```
[0 1 2 3 4]
```

5.4 Build ROCm JAX from source

Follow these steps to build JAX with ROCm support from source.

1. Clone the ROCm-specific fork of JAX with the desired branch:

   ```
   git clone https://github.com/ROCm/jax -b <branch_name>
   cd jax
   ```

2. Run the following command to build the necessary wheels:

   ```
   python3 ./build/build.py build --wheels=jaxlib,jax-rocm-plugin,jax-rocm-pjrt \
     --rocm_version=60 --rocm_path=/opt/rocm-[version]
   ```

   This will generate three wheels in the dist/ directory:

   - jaxlib (generic, device agnostic library)
   - jax-rocm-plugin (ROCm-specific plugin)
   - jax-rocm-pjrt (ROCm-specific runtime)

3. Install the custom JAX wheels.

   ```
   python3 setup.py develop --user && pip3 -m pip install dist/*.whl
   ```

5.4.1 Simplified build script

For a streamlined build process, consider using the jax/build/rocm/dev_build_rocm.py script. See https://github.com/rocm/jax/tree/main/build/rocm for more information.

5.5 Testing your JAX installation with ROCm

After launching the container, test whether JAX detects ROCm devices as expected:

```
python -c "import jax; print(jax.devices())"
python3 -c "import jax.numpy as jnp; x = jnp.arange(5); print(x)"
```

If the setup is successful, the output should list all available ROCm devices.

Expected output:

```
[RocmDevice(id=0), RocmDevice(id=1), RocmDevice(id=2), RocmDevice(id=3)]
```

```
[0 1 2 3 4]
```

CHAPTER

SIX

DGL ON ROCM

Deep Graph Library (DGL) is an easy-to-use, high-performance and scalable Python package for deep learning on graphs. DGL is framework agnostic, meaning if a deep graph model is a component in an end-to-end application, the rest of the logic is implemented using PyTorch.

For hardware, software, and third-party framework compatibility between ROCm and DGL, see the following resources:

- System requirements (Linux)
- DGL compatibility

6.1 Install DGL

To install DGL on ROCm, you have the following options:

- Use the prebuilt Docker image (recommended)
- Build your own docker image

6.1.1 Use a prebuilt Docker image with DGL pre-installed

The recommended way to set up a DGL environment and avoid potential installation issues is with Docker. The tested, prebuilt image includes DGL, PyTorch, ROCm, and other dependencies.

Important

To follow these instructions, input your chosen tag into <TAG>. Example: dgl-2.4_rocm6.4_ubuntu24.04_py3.12_pytorch_release_2.6.0.

You can download Docker images for DGL with specific ROCm, PyTorch, Python and operating system versions. See the available tags on Docker Hub and see docker image support below.

1. Download your required public DGL Docker image

```
docker pull rocm/dgl:<TAG>
```

2. Launch and connect to the Docker container using the image

```
docker run -it --cap-add=SYS_PTRACE --security-opt seccomp=unconfined \
--device=/dev/kfd --device=/dev/dri --group-add video \
--ipc=host --shm-size 8G rocm/dgl:<TAG>
```

> **Note**
>
> This will automatically download the image if it does not exist on the host. You can also pass the -v argument to mount any data directories from the host onto the container.

6.1.2 Docker image support

AMD validates and publishes ready-made DGL Docker images with ROCm backends on Docker Hub. The following Docker image tags and associated inventories are validated for ROCm 6.4.

PyTorch 2.6.0

Ubuntu 24.04

Tag
: rocm/dgl:dgl-2.4_rocm6.4_ubuntu24.04_py3.12_pytorch_release_2.6.0

Inventory

- ROCm 6.4.0
- Python 3.12.9
- PyTorch 2.6.0

PyTorch 2.4.1

Ubuntu 24.04

Tag
: rocm/dgl:dgl-2.4_rocm6.4_ubuntu24.04_py3.12_pytorch_release_2.4.1

Inventory

- ROCm 6.4.0
- Python 3.12.9
- PyTorch 2.4.1

Ubuntu 22.04

Tag
: rocm/dgl:dgl-2.4_rocm6.4_ubuntu22.04_py3.10_pytorch_release_2.4.1

Inventory

- ROCm 6.4.0
- Python 3.10.16
- PyTorch 2.4.1

PyTorch 2.3.0

Ubuntu 22.04

Tag
: rocm/dgl:dgl-2.4_rocm6.4_ubuntu22.04_py3.10_pytorch_release_2.3.0

Inventory

- ROCm 6.4.0

- Python 3.10.16
- PyTorch 2.3.0

6.1.3 Build your own Docker image

1. Clone the https://github.com/ROCm/dgl repository

```
git clone --recurse-submodules https://github.com/ROCm/dgl
cd dgl
```

2. Build the Docker container

DGL on Ubuntu 22.04 + ROCm 6.4 + Py 3.10 + PyTorch 2.4.1

To build the Docker container, run the following command:

```
# DGL on Ubuntu 22.04 + ROCm 6.4 + Py 3.10 + PyTorch 2.4.1
docker build \
  -t dgl:dgl-2.4_rocm6.4_ubuntu22.04_py3.10_pytorch_release_2.4.1 \
  --build-arg BASE_IMAGE=rocm/pytorch:rocm6.4_ubuntu22.04_py3.10_pytorch_release_2.4.1␣
↪\
  --build-arg ARG_CONDA_ENV=py_3.10 \
  --build-arg ARG_MAX_JOBS=8 \
  --build-arg ARG_GPU_BUILD_TARGETS="gfx90a,gfx942" \
  -f Dockerfile.rocm \
  .
```

DGL on Ubuntu 22.04 + ROCm 6.4 + Py 3.10 + PyTorch 2.3.0

To build the Docker container, run the following command:

```
# DGL on Ubuntu 22.04 + ROCm 6.4 + Py 3.10 + PyTorch 2.3.0
docker build \
  -t dgl:dgl-2.4_rocm6.4_ubuntu22.04_py3.10_pytorch_release_2.3.0 \
  --build-arg BASE_IMAGE=rocm/pytorch:rocm6.4_ubuntu22.04_py3.10_pytorch_release_2.3.0␣
↪\
  --build-arg ARG_CONDA_ENV=py_3.10 \
  --build-arg ARG_MAX_JOBS=8 \
  --build-arg ARG_GPU_BUILD_TARGETS="gfx90a,gfx942" \
  -f Dockerfile.rocm \
  .
```

DGL on Ubuntu 24.04 + ROCm 6.4 + Py 3.12 + PyTorch 2.4.1

To build the Docker container, run the following command:

```
# DGL on Ubuntu 24.04 + ROCm 6.4 + Py 3.12 + PyTorch 2.4.1
docker build \
  -t dgl:dgl-2.4_rocm6.4_ubuntu24.04_py3.12_pytorch_release_2.4.1 \
  --build-arg BASE_IMAGE=rocm/pytorch:rocm6.4_ubuntu24.04_py3.12_pytorch_release_2.4.1␣
↪\
  --build-arg ARG_CONDA_ENV=py_3.12 \
  --build-arg ARG_MAX_JOBS=8 \
```

(continues on next page)

(continued from previous page)

```
  --build-arg ARG_GPU_BUILD_TARGETS="gfx90a,gfx942" \
  -f Dockerfile.rocm \
  .
```

DGL on Ubuntu 24.04 + ROCm 6.4 + Py 3.12 + PyTorch 2.6.0

To build the Docker container, run the following command:

```
# DGL on Ubuntu 24.04 + ROCm 6.4 + Py 3.12 + PyTorch 2.6.0
docker build \
  -t dgl:dgl-2.4_rocm6.4_ubuntu24.04_py3.12_pytorch_release_2.6.0 \
  --build-arg BASE_IMAGE=rocm/pytorch:rocm6.4_ubuntu24.04_py3.12_pytorch_release_2.6.0␣
↪\
  --build-arg ARG_CONDA_ENV=py_3.12 \
  --build-arg ARG_MAX_JOBS=8 \
  --build-arg ARG_GPU_BUILD_TARGETS="gfx90a,gfx942" \
  -f Dockerfile.rocm \
  .
```

6.2 Test the DGL installation

To verify that DGL has been successfully installed, run the Docker container as described in the installing DGL section. Once inside the container, ensure you have access to the Bash shell.

To check for a shared library:

```
find / -name libdgl.so -print -quit 2>/dev/null && echo "libdgl.so found" || echo "libdgl.so NOT found"
```

To check for Python import:

```
conda activate py_<python version> #You can check this with conda info --envs

export DGLBACKEND=pytorch

python -c "import dgl; print('dgl import successful, version:', dgl.__version__)" || echo "Failed to␣
↪import DGL"
```

6.3 Run a DGL example

Multiple use cases of DGL have been tested and verified. However, a recommended example follows a drug discovery pipeline using the SE3Transformer. This detailed procedure and steps will be outlined in the AMD ROCm blog, where you can search for DGL examples.

6.4 Troubleshooting

- Unable to access Docker or GPU in user accounts? Ensure the user is added to docker, video, and render groups. See Configuring permissions for GPU access.
- Profiling DGL workloads? Use the PyTorch Profiler, as explained in PyTorch Profiler to profile GPU kernels on ROCm.

CHAPTER

SEVEN

RUNNING ROCM DOCKER CONTAINERS

Using Docker to run your ROCm applications is one of the best ways to get consistent and reproducible environments.

7.1 Prerequisites

- amdgpu-dkms: Docker containers share the kernel with the host OS. Therefore, the ROCm kernel-mode driver (amdgpu-dkms) must be installed on the host. If you've already installed ROCm, you probably already have amdgpu-dkms.
  - Check for amdgpu-dkms
  - If you don't have amdgpu-dkms, follow the standard install instructions (which comes with amdgpu-dkms) or install amdgpu-dkms only.

> **See also**
>
> For instructions on installing Docker, see the official Docker installation documentation.

7.2 Accessing GPUs in containers

To grant a Docker container access to the host's AMD GPUs, run your container with the following options. See the Docker documentation to learn more about the docker run command and its options.

```
docker run --device /dev/kfd --device /dev/dri --security-opt seccomp=unconfined <image>
```

The purpose of each option is as follows:

- --device /dev/kfd

  This is the main compute interface, shared by all GPUs. The Docker CLI's --device option enables directly exposing host devices to a container. See Add host device to container (–device) for more information.

- --device /dev/dri

  This directory contains the Direct Rendering Interface (DRI) for each GPU. To restrict access to specific GPUs, see Restricting GPU access.

- --security-opt seccomp=unconfined (optional)

  This option enables memory mapping, and is recommended for containers running in HPC environments. See Optional security options (–security-opt).

The performance of an application can vary depending on the assignment of GPUs and CPUs to the task. Typically, numactl is installed as part of many HPC applications to provide GPU/CPU mappings. This Docker runtime option supports memory mapping and can improve performance.

7.2.1 Docker compose

You can also use docker compose to launch your containers, even when launching a single container. This can be a convenient way to run complex Docker commands without having to remember all the CLI arguments. The following snippet is an example compose.yaml file, which is equivalent to the preceding docker run command:

```
services:
  my-service:
    image: <image>
    devices:
      - /dev/kfd
      - /dev/dri
    security_opt:
      - seccomp=unconfined
```

You can then run this using docker compose run my-service.

7.2.2 Restricting GPU access

By default, passing --device /dev/dri grants access to all GPUs on the system. To limit a container to a specific subset of GPUs, you can instead pass in their individual device nodes.

GPU device nodes are located in /dev/dri/ and are typically named renderD128, renderD129, and so on. You can list the available GPUs on your host system with the following command:

```
ls /dev/dri/render*
```

To expose only the first two GPUs to the container, specify them directly in the run command. Note that /dev/kfd is always required for the compute interface.

For example, to expose the first and second GPU:

```
docker run --device /dev/kfd --device /dev/dri/renderD128 --device /dev/dri/renderD129 ..
```

7.2.3 Verifying the amdgpu driver has been loaded on GPUs

rocminfo is an application for reporting information about the HSA system attributes and agents. amd-smi is a tool that acts as a command line interface for manipulating and monitoring the amdgpu kernel.

Running rocminfo and amd-smi list inside the container will only enumerate the GPUs passed into the docker container. Running rocminfo and amd-smi list on bare metal will enumerate all ROCm-capable GPUs on the machine.

7.3 Docker images in the ROCm ecosystem

The ROCm Docker repository hosts Dockerfiles useful for building your own ROCm-capable containers. The built images are available on Docker Hub. In particular:

- rocm/rocm-terminal is a small image with the prerequisites to build HIP applications, but does not include any libraries.

- ROCm dev images provide a variety of OS + ROCm versions, and are a great starting place for building applications.

7.3.1 Applications

AMD provides pre-built images for various GPU-ready AI and HPC applications through Infinity Hub. There, you'll also find examples for invoking each application and suggested parameters used for benchmarking.

CHAPTER

EIGHT

USING SPACK TO INSTALL ROCM PACKAGES

Spack is a package management tool designed to support multiple software versions and configurations on a wide variety of platforms and environments. It was designed for large supercomputing centers, where many users share common software installations on clusters with exotic architectures using libraries that do not have a standard ABI. Spack is non-destructive: installing a new version does not break existing installations, so many configurations can coexist on the same system.

Most importantly, Spack is simple. It offers a simple spec syntax, so users can concisely specify versions and configuration options. Spack is also simple for package authors: package files are written in pure Python, and specs allow package authors to maintain a single file for many different builds of the same package.

See the official Spack documentation for more information.

8.1 Installing prerequisites for Spack

> **Note**
>
> You must install all prerequisites before installing Spack.

Ubuntu

```
# Install some essential utilities:
apt-get update
apt-get install make patch bash tar gzip unzip bzip2 file gnupg2 git gawk
apt-get update -y
apt-get install -y xz-utils
apt-get install build-essential
apt-get install vim
apt-get install libpci-dev
# Install Python:
apt-get install python3
apt-get upgrade python3-pip
# Install Compilers:
apt-get install gcc
apt-get install gfortran
```

SLES

```
# Install some essential utilities:
zypper update
zypper install make patch bash tar gzip unzip bzip xz file gnupg2 git awk
zypper in -t pattern
zypper install vim
# Install Python:
zypper install python3
zypper install python3-pip
# Install Compilers:
zypper install gcc
zypper install gcc-fortran
zypper install gcc-c++
```

8.2 Building ROCm components using Spack

1. To use the Spack package manager, clone the Spack project from https://github.com/spack/spack.

   ```
   git clone https://github.com/spack/spack.git
   ```

2. Initialize Spack.

   The setup-env.sh script initializes the Spack environment.

   ```
   cd spack
   . share/spack/setup-env.sh
   ```

   Spack commands are available once the above steps are completed. To list the available commands, use help.

   ```
   spack help
   ```

8.3 ROCm packages in Spack

Component	Spack package name	Minimum supported version	Latest supported version
AMD SMI	amdsmi	5.5.0	6.4.1
aqlprofile	aqlprofile	5.5.0	6.4.1
comgr	comgr	5.5.0	6.4.1
Composable Kernel	composable-kernel	5.5.0	6.4.1
devicelibs	rocm-device-libs	5.5.0	6.4.1
HIP (hip_in_vdi)	hip	5.5.0	6.4.1
hipBLAS	hipblas	5.5.0	6.4.1
hipBLASLt	hipblaslt	6.0.0	6.4.1
HIPCC	hipcc	5.7.0	6.4.1
hipCUB	hipcub	5.5.0	6.4.1
hipFFT	hipfft	5.5.0	6.4.1
hipfort	hipfort	5.5.0	6.4.1
HIPIFY	hipify-clang	5.5.0	6.4.1
hipRAND	hiprand	5.5.0	6.4.1

continues on next page

Table 8.1 – continued from previous page

Component	Spack package name	Minimum supported version	Latest supported version
hipSOLVER	hipsolver	5.5.0	6.4.1
hipSPARSE	hipsparse	5.5.0	6.4.1
hipSPARSELt	hipsparselt	6.0.0	6.4.1
hipTensor	hip-tensor	5.7.0	6.4.1
HIP Tests	hip-tests	6.1.0	6.4.1
lightning	llvm-amdgpu	5.5.0	6.4.1
MIOpen (HIP)	miopen-hip	5.5.0	6.4.1
MIGraphX	migraphx	5.5.0	6.4.1
MIVisionX	mivisionx	5.5.0	6.4.1
OpenCL	rocm-opencl	5.5.0	6.4.1
openmp-extras	rocm-openmp-extras	5.5.0	6.4.1
RCCL	rccl	5.5.0	6.4.1
rocAL	rocal	6.2.0	6.4.1
rocALUTION	rocalution	5.5.0	6.4.1
rocBLAS	rocblas	5.5.0	6.4.1
ROCdbgapi	rocm-dbgapi	5.5.0	6.4.1
rocDecode	rocdecode	6.1.0	6.4.1
rocFFT	rocfft	5.5.0	6.4.1
rocJPEG	rocjpeg	6.3.0	6.4.1
rocm-core	rocm-core	5.5.0	6.4.1
rocminfo	rocminfo	5.5.0	6.4.1
rocMLIR	rocmlir	5.4.0	6.4.1
ROCm Bandwidth Test	rocm-bandwidth-test	5.5.0	6.4.1
rocm-cmake	rocm-cmake	5.5.0	6.4.1
ROCm Compute Profiler	rocprofiler-compute	6.3.2	6.4.1
ROCm Data Center Tool (RDC)	rdc	5.5.0	6.4.1
ROCm Debug Agent	rocm-debug-agent	5.5.0	6.4.1
ROCm Debugger (ROCgdb)	rocm-gdb	5.5.0	6.4.1
ROCm Examples	rocm-examples	6.2.0	6.4.1
ROCm SMI Library	rocm-smi-lib	5.5.0	6.4.1
ROCm Systems Profiler	rocprofiler-systems	6.3.0	6.4.1
ROCm Validation Suite	rocm-validation-suite	5.5.0	6.4.1
rocPRIM	rocprim	5.5.0	6.4.1
ROCProfiler	rocprofiler-dev	5.5.0	6.4.1
rocprofiler-register	rocprofiler-register	6.1.0	6.4.1
ROCprofiler-SDK	rocprofiler-sdk	6.2.4	6.4.1
rocPyDecode	rocpydecode	6.2.0	6.4.1
rocRAND	rocrand	5.5.0	6.4.1
ROCr Runtime	hsa-rocr-dev	5.5.0	6.4.1
rocSHMEM	rocshmem	6.4.0	6.4.1
rocSOLVER	rocsolver	5.5.0	6.4.1
rocSPARSE	rocsparse	5.5.0	6.4.1
rocThrust	rocthrust	5.5.0	6.4.1
ROCTracer	roctracer-dev	5.5.0	6.4.1
roctracer-dev-api	roctracer-dev-api	5.5.0	6.4.1
rocWMMA	rocwmma	5.5.0	6.4.1
ROCm Performance Primitives (RPP)	rpp	5.7.0	6.4.1
Tensile	rocm-tensile	5.5.0	6.4.1
TransferBench	transferbench	6.3.0	6.4.1
atmi	atmi	5.5.0	5.5.1 (final)

continues on next page

Table 8.1 – continued from previous page

Component	Spack package name	Minimum supported version	Latest supported version
clang-ocl	rocm-clang-ocl	5.5.0	6.1.2 (final)
mlirmiopen	mlirmiopen	5.3.0	5.4.0 (deprecated)
MIOpen (GEMM)	miopengemm	5.5.0	5.5.1 (final)
MIOpen (OpenCL)	miopen-opencl	5.5.0	5.5.1 (final)
Omniperf	omniperf	6.2.0	6.3.1 (final)
Omnitrace	omnitrace	rocm-6.2.0	rocm-6.3.0 (final)
rocclr (vdi)	hip-rocclr	5.5.0	5.6.1 (final)
ROCT Thunk Interface	hsakmt-roct	5.5.0	6.2.4 (final)

8.4 Installing ROCm components using Spack

1. rocm-cmake

   Install the default variants and the latest version of rocm-cmake.

   ```
   spack install rocm-cmake
   ```

   To install a specific version of rocm-cmake, use:

   ```
   spack install rocm-cmake@<version number>
   ```

   For example, spack install rocm-cmake@6.4.1

2. info

   The info command displays basic package information. It shows the preferred, safe, and deprecated versions, in addition to the available variants. It also shows the dependencies with other packages.

   ```
   spack info mivisionx
   ```

   For example:

   ```
   $ spack info mivisionx
   CMakePackage:   mivisionx

   Description:
       MIVisionX toolkit is a set of comprehensive computer vision and machine
       intelligence libraries, utilities, and applications bundled into a
       single toolkit.

   Homepage: https://github.com/ROCm/MIVisionX

   Preferred version:
       6.4.1    https://github.com/ROCm/MIVisionX/archive/rocm-6.4.1.tar.gz

   Safe versions:
       6.4.1    https://github.com/ROCm/MIVisionX/archive/rocm-6.4.1.tar.gz
       6.4.0    https://github.com/ROCm/MIVisionX/archive/rocm-6.4.0.tar.gz
       6.3.3    https://github.com/ROCm/MIVisionX/archive/rocm-6.3.3.tar.gz
       6.3.2    https://github.com/ROCm/MIVisionX/archive/rocm-6.3.2.tar.gz
       6.3.1    https://github.com/ROCm/MIVisionX/archive/rocm-6.3.1.tar.gz
       6.3.0    https://github.com/ROCm/MIVisionX/archive/rocm-6.3.0.tar.gz
   ```

(continues on next page)

(continued from previous page)

```
    6.2.4    https://github.com/ROCm/MIVisionX/archive/rocm-6.2.4.tar.gz
    6.2.1    https://github.com/ROCm/MIVisionX/archive/rocm-6.2.1.tar.gz
    6.2.0    https://github.com/ROCm/MIVisionX/archive/rocm-6.2.0.tar.gz
    6.1.2    https://github.com/ROCm/MIVisionX/archive/rocm-6.1.2.tar.gz
    6.1.1    https://github.com/ROCm/MIVisionX/archive/rocm-6.1.1.tar.gz
    6.1.0    https://github.com/ROCm/MIVisionX/archive/rocm-6.1.0.tar.gz
    6.0.2    https://github.com/ROCm/MIVisionX/archive/rocm-6.0.2.tar.gz
    6.0.0    https://github.com/ROCm/MIVisionX/archive/rocm-6.0.0.tar.gz
    5.7.1    https://github.com/ROCm/MIVisionX/archive/rocm-5.7.1.tar.gz
    5.7.0    https://github.com/ROCm/MIVisionX/archive/rocm-5.7.0.tar.gz
    5.6.1    https://github.com/ROCm/MIVisionX/archive/rocm-5.6.1.tar.gz
    5.6.0    https://github.com/ROCm/MIVisionX/archive/rocm-5.6.0.tar.gz
    5.5.1    https://github.com/ROCm/MIVisionX/archive/rocm-5.5.1.tar.gz
    5.5.0    https://github.com/ROCm/MIVisionX/archive/rocm-5.5.0.tar.gz

Deprecated versions:
    5.4.3    https://github.com/ROCm/MIVisionX/archive/rocm-5.4.3.tar.gz
    5.4.0    https://github.com/ROCm/MIVisionX/archive/rocm-5.4.0.tar.gz
    5.3.3    https://github.com/ROCm/MIVisionX/archive/rocm-5.3.3.tar.gz
    5.3.0    https://github.com/ROCm/MIVisionX/archive/rocm-5.3.0.tar.gz

Variants:
    add_tests [false]            false, true
        add tests and samples folder
    asan [false]                 false, true
        Build with address-sanitizer enabled or disabled
    build_system [cmake]         cmake
        Build systems supported by the package
    hip [true]                   false, true
        Use HIP as backend
    opencl [false]               false, true
        Use OPENCL as the backend

    when build_system=cmake
      build_type [Release]       Debug, MinSizeRel, RelWithDebInfo, Release
          CMake build type
      generator [make]           none
          the build system generator to use

    when build_system=cmake ^cmake@3.9:
      ipo [false]                false, true
          CMake interprocedural optimization

Build Dependencies:
    cmake   hip             miopen-hip      opencv      py-google-api-python-client  py-pytz         rapidjson
    cxx     libjpeg-turbo   miopen-opencl   openssl     py-numpy                     py-setuptools   rocm-
↪core
    ffmpeg  lmdb            miopengemm      protobuf    py-protobuf                  py-wheel        rocm-
↪opencl
    gmake   migraphx        ninja           py-future   py-pybind11                  python          rpp

Link Dependencies:
```

(continues on next page)

(continued from previous page)

```
    hip       miopen-hip      openssl                    py-numpy      py-setuptools  rocm-core
    lmdb      miopen-opencl   py-future                  py-pybind11   py-wheel       rocm-opencl
    migraphx  miopengemm      py-google-api-python-client  py-pytz      rapidjson      rpp

Run Dependencies:
    py-protobuf

Licenses:
    MIT
```

8.5 Installing variants for ROCm components

The variants listed above indicate that the mivisionx package is built by default with build_type=Release and the hip backend, and without the opencl backend. build_type=Debug and RelWithDebInfo, with opencl and without hip, are also supported.

For example:

```
spack install mivisionx build_type=Debug #Backend will be hip since it is the default one
spack install mivisionx+opencl build_type=Debug #Backend will be opencl and hip will be disabled as␣
↪per the conflict defined in recipe
```

- spack spec command

  To display the dependency tree, the spack spec command can be used with the same format.

  For example:

```
$ spack spec mivisionx

-   mivisionx@6.4.1~add_tests~asan+hip~ipo~opencl build_system=cmake build_type=Release␣
↪generator=make arch=linux-ubuntu22.04-zen2 %cxx=gcc@11.4.0
-       ^cmake@3.31.8~doc+ncurses+ownlibs~qtgui build_system=generic build_type=Release␣
↪arch=linux-ubuntu22.04-zen2 %c,cxx=gcc@11.4.0
-           ^curl@8.11.1~gssapi~ldap~libidn2~librtmp~libssh~libssh2+nghttp2 build_
↪system=autotools libs:=shared,static tls:=openssl arch=linux-ubuntu22.04-zen2 %c,cxx=gcc@11.4.
↪0
-               ^nghttp2@1.65.0 build_system=autotools arch=linux-ubuntu22.04-zen2 %c,
↪cxx=gcc@11.4.0
-           ^ncurses@6.5~symlinks+termlib abi=none build_system=autotools patches:=7a351bc␣
↪arch=linux-ubuntu22.04-zen2 %c,cxx=gcc@11.4.0
...
```

8.6 Creating an environment

You can create an environment with all the required components of your version.

1. In the root folder, create a new folder when you can create a .yaml file. This file is used to create an environment.

```
mkdir /localscratch
cd /localscratch
vi sample.yaml
```

2. Add all the required components in the sample.yaml file. For example:

```
spack:
concretization: separately
packages:
all:
compiler: [gcc@8.5.0]
specs:
- matrix:
- ['%gcc@8.5.0\^cmake@3.19.7']
- [rocm-cmake@6.4.1, rocm-dbgapi@6.4.1, rocm-debug-agent@6.4.1, rocm-gdb@6.4.1,
  rocminfo@6.4.0, rocm-opencl@6.4.0, rocm-smi-lib@6.4.0, rocm-tensile@6.4.0, rocm-validation-
↪suite@6.4.1,
  rocprim@6.4.1, rocprofiler-dev@6.4.1, rocrand@6.4.1, rocsolver@6.4.1, rocsparse@6.4.1,
  rocthrust@6.4.1, roctracer-dev@6.4.1]
view: true
```

3. Once you've created the .yaml file, you can use it to create an environment.

```
spack env create -d /localscratch/MyEnvironment /localscratch/sample.yaml
```

4. Activate the created environment.

```
spack env activate /localscratch/MyEnvironment
```

5. Before installing, verify that you want all the component versions.

```
spack find # this command will list out all components been in the environment (and 0 installed )
```

6. Install all the components in the .yaml file.

```
cd /localscratch/MyEnvironment
spack install -j 50
```

7. Check that all components are successfully installed.

```
spack find
```

8. If any modification is made to the .yaml file, you must deactivate the existing environment and create a new one in order for the modifications to be reflected.

   To deactivate, use:

```
spack env deactivate
```

8.7 Creating and applying a patch before installation

Spack installs ROCm packages after pulling the source code from GitHub and building it locally. In order to build a component with any modification to the source code, you must generate a patch and apply it before the build phase.

To generate a patch and build with the changes:

1. Stage the source code. For example:

```
spack stage hip@6.4.1
# (This will pull the 6.4.1 release version source code of hip and display the path to spack-src␣
↪directory where entire source code is available)
```

You should see something like this:

```
==> Using cached archive: /data/root/temp/rocm-6.4.1/spack/var/spack/cache/_source-cache/
↪archive/d8/d8dba8cdf05463afb7879de2833983cafa6a006ba719815a35b96d9b92fc7fc4.tar.gz
==> Using cached archive: /data/root/temp/rocm-6.4.1/spack/var/spack/cache/_source-cache/
↪archive/82/829e61a5c54d0c8325d02b0191c0c8254b5740e63b8bfdb05eec9e03d48f7d2c.tar.gz
==> Using cached archive: /data/root/temp/rocm-6.4.1/spack/var/spack/cache/_source-cache/
↪archive/80/8081d4ab1a43ffa1cebd646668d83008b799ab98c14daf7b455922355a439c8a.tar.gz
==> Moving resource stage
        source: /tmp/root/spack-stage/resource-clr-zo53ondw3tevsr3gmoofbhre7asvis46/spack-src/
        destination: /tmp/root/spack-stage/spack-stage-hip-6.4.1-zo53ondw3tevsr3gmoofbhre7asvis46/
↪spack-src/clr
==> Moving resource stage
        source: /tmp/root/spack-stage/resource-hip-tests-zo53ondw3tevsr3gmoofbhre7asvis46/spack-
↪src/
        destination: /tmp/root/spack-stage/spack-stage-hip-6.4.1-zo53ondw3tevsr3gmoofbhre7asvis46/
↪spack-src/hip-tests
==> Staged hip in /tmp/root/spack-stage/spack-stage-hip-6.4.1-zo53ondw3tevsr3gmoofbhre7asvis46
```

2. Change directory to spack-src inside the staged directory.

```
root@computername:/spack$ cd /tmp/root/spack-stage/spack-stage-hip-6.4.1-
↪zo53ondw3tevsr3gmoofbhre7asvis46
root@computername:/tmp/root/spack-stage/spack-stage-hip-6.4.1-
↪zo53ondw3tevsr3gmoofbhre7asvis46$ cd spack-src/
```

3. Create a new Git repository.

```
root@computername:/tmp/root/spack-stage/spack-stage-hip-6.4.1-
↪zo53ondw3tevsr3gmoofbhre7asvis46/spack-src$ git init
```

4. Add the entire directory to the repository.

```
root@computername:/tmp/root/spack-stage/spack-stage-hip-6.4.1-
↪zo53ondw3tevsr3gmoofbhre7asvis46/spack-src$ git add .
```

5. Make the required changes to the source code.

```
root@computername:/tmp/root/spack-stage/spack-stage-hip-6.4.1-
↪zo53ondw3tevsr3gmoofbhre7asvis46/spack-src# vi hipamd/CMakeLists.txt
# Make required changes in the source code
```

6. Generate the patch using the git diff command.

```
diff > /spack/var/spack/repos/builtin/packages/hip/0001-modifications.patch
root@computername:/tmp/root/spack-stage/spack-stage-hip-6.4.1-
↪zo53ondw3tevsr3gmoofbhre7asvis46/spack-src$ git diff > /spack/var/spack/repos/builtin/
↪packages/hip/0001-modifications.patch
```

7. Update the recipe with the patch file name and any conditions you want to apply.

```
root@computername:/tmp/root/spack-stage/spack-stage-hip-6.4.1-
↪zo53ondw3tevsr3gmoofbhre7asvis46/spack-src$ spack edit hip
```

8. Provide the patch file name and the conditions for the patch to be applied in the hip recipe as follows.

```
patch("0001-modifications.patch", when="@6.4.1")
```

   Spack will apply 0001-modifications.patch on the 6.4.1 release code before starting the hip build.

9. After each modification, you must update the recipe. If there is no change to the recipe, run

```
touch /spack/var/spack/repos/builtin/packages/hip/package.py
```

CHAPTER

NINE

PACKAGE MANAGER INTEGRATION

This section provides information about the required meta-packages for the following AMD ROCm programming models:

- Heterogeneous-Computing Interface for Portability (HIP)
- OpenCL™
- OpenMP™

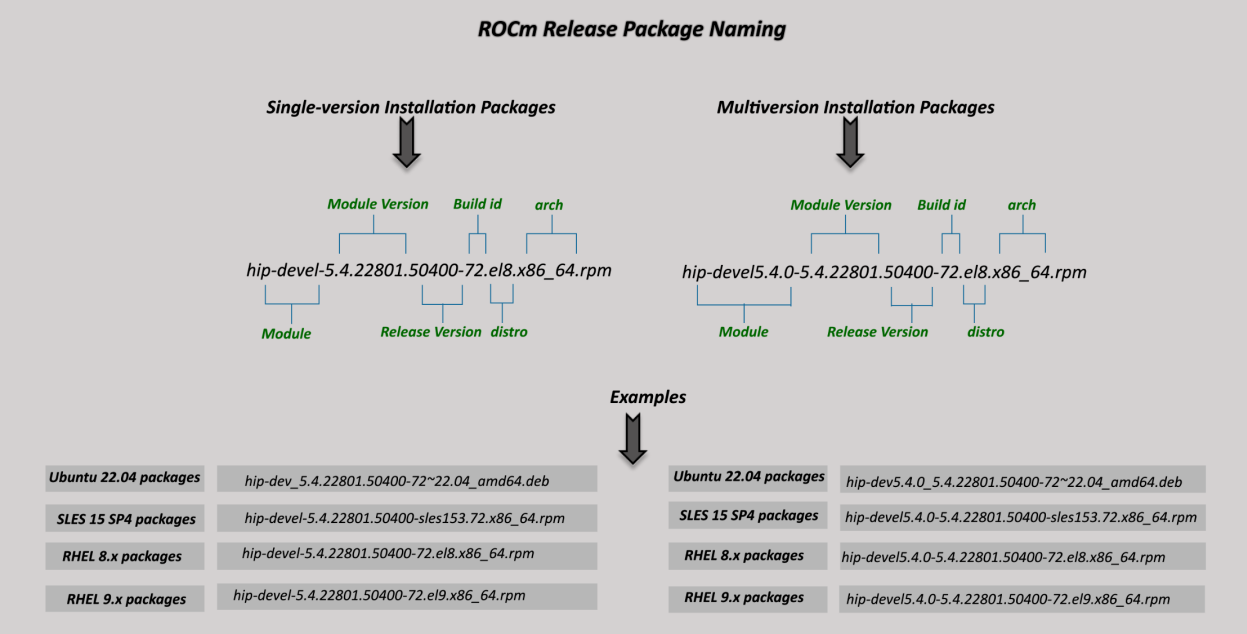
9.1 ROCm package naming conventions

A meta-package is a grouping of related packages and dependencies used to support a specific use case.

Example: Running HIP applications

All meta-packages exist in both versioned and non-versioned forms.

- Non-versioned packages: For a single-version installation of the ROCm stack
- Versioned packages: For multi-version installations of the ROCm stack



The figure above demonstrates the single and multi-version ROCm packages' naming structure, including examples for various Linux distributions. See terms below:

Module - It is the part of the package that represents the name of the ROCm component.

Example: The examples mentioned in the image represent the ROCm HIP module.

Module version - It is the version of the library released in that package. It should increase with a newer release.

Release version - It shows the ROCm release version when the package was released.

Example: 50400 points to the ROCm 5.4.0 release.

Build id - It represents the build number for that release.

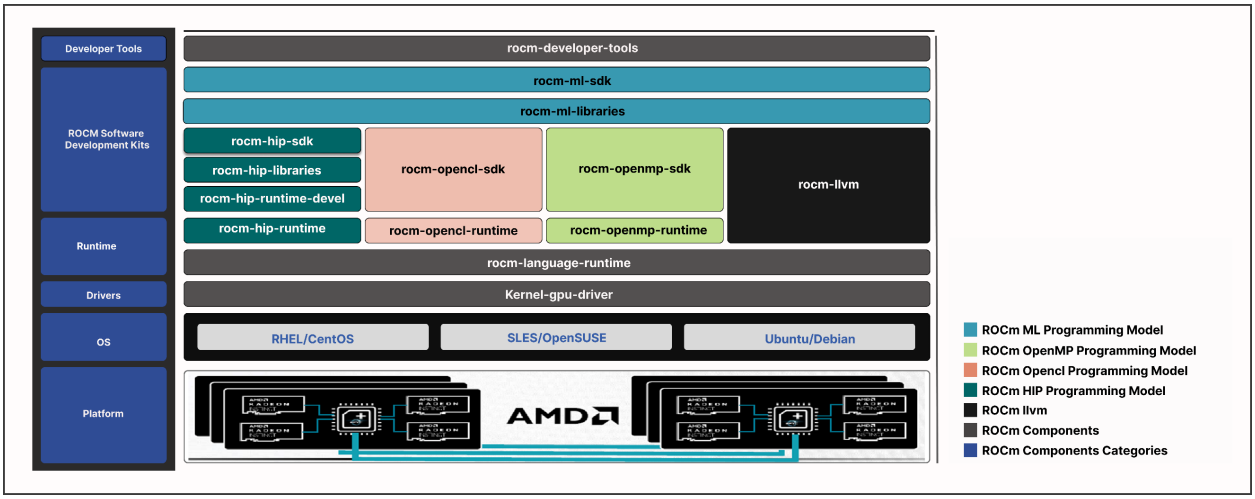
Arch - It shows the architecture for which the package was created.

Distro - It describes the distribution for which the package was created. It is valid only for rpm packages.

Example: el8 represents RHEL 8.x packages.

9.2 Components of ROCm programming models

The figure below demonstrates the high-level layered architecture of ROCm programming models and their meta-packages. All meta-packages are a combination of required packages and libraries.



> **Note**
>
> The preceding figure is for informational purposes only. The individual packages in a meta-package are subject to change. To avoid conflicts, install meta-packages, not individual packages.

Example:

- rocm-hip-runtime is used to deploy on supported machines to execute HIP applications.
- rocm-hip-sdk contains runtime components to deploy and execute HIP applications.

> **Note**
>
> rocm-llvm is not a meta-package; it's a single package that installs the ROCm Clang compiler files.

Package	Description
rocm	All ROCm core packages, tools, and libraries.
rocm-language-runtime	The ROCm runtime.
rocm-developer-tools	Debug and profile HIP applications.
rocm-hip-runtime	Run HIP applications writen for the AMD platform.
rocm-hip-runtime-devel	Develop applications on HIP or port from CUDA.
rocm-opencl-runtime	Run OpenCL-based applications on the AMD platform.
rocm-opencl-sdk	Develop OpenCL-based applications for the AMD platform.
rocm-hip-libraries	HIP libraries optimized for the AMD platform.
rocm-hip-sdk	Develop or port HIP applications and libraries for the AMD platform.
rocm-ml-libraries	Key machine learning libraries. Includes MIOpen.
rocm-ml-sdk	Develop and run machine learning applications for AMD.
rocm-openmp-runtime	Run OpenMP-based applications on the AMD platform.
rocm-openmp-sdk	Develop OpenMP-based applications for the AMD software.

9.3 Packages in ROCm programming models

This section discusses the available meta-packages and their packages. The following table shows the meta-packages and their associated (meta-)packages in a ROCm programming model.

> **Note**
>
> rocm-utils is a legacy meta-package that includes rocminfo, rocm-cmake, and rocm-core. Other meta-packages manage the installation of these components.

Meta package	Associated packages
rocm	Meta packages: rocm-developer-tools, rocm-ml-sdk, rocm-opencl-sdk, rocm-openmp-sdk Packages: migraphx, migraphx-devel, mivisionx, rocm-core, rpp-devel
rocm-developer-tools	Meta packages: rocm-language-runtime Packages: amd-smi-lib, hsa-amd-aqlprofile, rocm-core, rocm-dbgapi, rocm-debug-agent, rocm-gdb, rocm-smi-lib, rocprofiler, rocprofiler-devel, rocprofiler-compute, rocprofiler-plugins, rocprofiler-register, rocprofiler-sdk, rocprofiler-sdk-roctx, rocprofiler-systems, roctracer, roctracer-devel
rocm-ml-sdk	Meta packages: rocm-hip-sdk, rocm-ml-libraries Packages: miopen-hip-devel, rocm-core
rocm-ml-libraries	Meta packages: rocm-hip-libraries Packages: miopen-hip, rocm-core, rocm-llvm
rocm-hip-sdk	Meta packages: rocm-hip-libraries, rocm-hip-runtime-devel Packages: composablekernel-devel, hipblas-common-devel, hipblas-devel, hipblaslt-devel, hipcub-devel, hipfft-devel, hipfort-devel, hiprand-devel, hipsolver-devel, hipsparse-devel, hipsparselt-devel, hiptensor-devel, rccl-devel, rocalution-devel, rocblas-devel, rocfft-devel, rocm-core, rocprim-devel, rocrand-devel, rocsolver-devel, rocsparse-devel, rocthrust-devel, rocwmma-devel
rocm-hip-libraries	Meta packages: rocm-hip-runtime Packages: hipblas, hipblaslt, hipfft, hiprand, hipsolver, hipsparse, hipsparselt, hiptensor, rccl, rocalution, rocblas, rocfft, rocm-core, rocm-smi-lib, rocrand, rocsolver, rocsparse
rocm-hip-runtime-devel	Meta packages: rocm-hip-runtime Packages: hipcc, hip-doc, hip-devel, hip-samples, hipify-clang, hsa-rocr-devel, rocm-cmake, rocm-core, rocm-device-libs, rocm-llvm
rocm-hip-runtime	Meta packages: rocm-language-runtime Packages: hip-runtime-amd, rocm-core, rocminfo
rocm-language-runtime	Packages: comgr, hsa-rocr, openmp-extras-runtime, rocm-core
rocm-openmp-sdk	Meta packages: rocm-language-runtime Packages: hsa-rocr-devel, openmp-extras-devel, rocm-core, rocm-device-libs, rocm-llvm
rocm-opencl-sdk	Meta packages: rocm-opencl-runtime Packages: hsa-rocr-devel, rocm-core, rocm-opencl-devel
rocm-opencl-runtime	Meta packages: rocm-language-runtime Packages:

CHAPTER

TEN

SYSTEM REQUIREMENTS (LINUX)

10.1 Supported GPUs

The following table shows the supported AMD Instinct™ accelerators, and Radeon™ PRO and Radeon GPUs. If a GPU is not listed on this table, it's not officially supported by AMD.

Accelerators and GPUs listed in the following table support compute workloads (no display information or graphics). If you're using ROCm with AMD Radeon or Radeon PRO GPUs for graphics workloads, see the Use ROCm on Radeon GPU documentation to verify compatibility and system requirements.

AMD Instinct

Accelerator	Architecture	LLVM target	Support
AMD Instinct MI325X	CDNA3	gfx942	[1]
AMD Instinct MI300X	CDNA3	gfx942	
AMD Instinct MI300A	CDNA3	gfx942	
AMD Instinct MI250X	CDNA2	gfx90a	
AMD Instinct MI250	CDNA2	gfx90a	
AMD Instinct MI210	CDNA2	gfx90a	
AMD Instinct MI100	CDNA	gfx908	
AMD Instinct MI50	GCN5.1	gfx906	
AMD Instinct MI25	GCN5.0	gfx900	

AMD Radeon PRO

GPU	Architecture	LLVM target	Support
AMD Radeon AI PRO R9700	RDNA4	gfx1201	[5]
AMD Radeon PRO V710	RDNA3	gfx1101	
AMD Radeon PRO W7900 Dual Slot	RDNA3	gfx1100	
AMD Radeon PRO W7900	RDNA3	gfx1100	
AMD Radeon PRO W7800 48GB	RDNA3	gfx1100	
AMD Radeon PRO W7800	RDNA3	gfx1100	
AMD Radeon PRO W7700	RDNA3	gfx1101	Page 150, 5
AMD Radeon PRO W6800	RDNA2	gfx1030	
AMD Radeon PRO V620	RDNA2	gfx1030	
AMD Radeon PRO VII	GCN5.1	gfx906	

[1] AMD Instinct MI325X is supported only on Ubuntu 22.04 [GA Kernel 5.15].

AMD Radeon

GPU	Architecture	LLVM target	Support
AMD Radeon RX 9070 XT	RDNA4	gfx1201	[5]
AMD Radeon RX 9070 GRE	RDNA4	gfx1201	[5]
AMD Radeon RX 9070	RDNA4	gfx1201	[5]
AMD Radeon RX 9060 XT	RDNA4	gfx1200	[5]
AMD Radeon RX 7900 XTX	RDNA3	gfx1100	
AMD Radeon RX 7900 XT	RDNA3	gfx1100	
AMD Radeon RX 7900 GRE	RDNA3	gfx1100	[5]
AMD Radeon RX 7800 XT	RDNA3	gfx1101	[5]
AMD Radeon VII	GCN5.1	gfx906	

: Supported - Official software distributions of the current ROCm release fully support this hardware.

: Deprecated - The current ROCm release has limited support for this hardware. Existing features and capabilities are maintained, but no new features or optimizations will be added. A future ROCm release will remove support.

: Unsupported - The current ROCm release does not support this hardware. The HIP runtime might continue to run applications for an unsupported GPU, but prebuilt ROCm libraries are not officially supported and will cause runtime errors.

Important

Systems with multiple GPUs may require iommu=pt to be set at boot time to prevent application hangs, as described in Issue #5: Application hangs on Multi-GPU systems.

Note

See the Compatibility matrix for an overview of supported GPU architectures across ROCm releases.

10.2 Supported operating systems

AMD ROCm software supports the following Linux distributions.

[5] Radeon AI PRO R9700, Radeon RX 9070, Radeon RX 9070 GRE, Radeon RX 9070 XT, Radeon RX 9060 XT, Radeon PRO W7700, and Radeon RX 7800 XT are supported only on Ubuntu 24.04.2, Ubuntu 22.04.5, RHEL 9.6, RHEL 9.5, and RHEL 9.4.

Operating system	Kernel	Glibc	Support
Ubuntu 24.04.2	6.8 [GA], 6.11 [HWE]	2.39	
Ubuntu 22.04.5	5.15 [GA], 6.8 [HWE]	2.35	
RHEL 9.6	5.14+	2.34	
RHEL 9.5	5.14+	2.34	
RHEL 9.4	5.14+	2.34	
RHEL 8.10	4.18.0+	2.28	
SLES 15 SP6	6.5.0+	2.38	
Oracle Linux 9	5.15.0 (UEK)	2.35	[2]
Oracle Linux 8	5.15.0 (UEK)	2.28	[2]
Azure Linux 3.0	6.6.60	2.38	[3]
Debian 12	6.1	2.36	[4]

Note

- See Red Hat Enterprise Linux Release Dates to learn about the specific kernel versions supported on Red Hat Enterprise Linux (RHEL).
- See List of SUSE Linux Enterprise Server kernel to learn about the specific kernel version supported on SUSE Linux Enterprise Server (SLES).
- See the Compatibility matrix for an overview of OS support across ROCm releases.

10.3 Virtualization support

ROCm supports virtualization for the Instinct accelerators and Radeon PRO GPUs listed in the following table.

Note

These virtualization technologies are designed to dedicate entire GPUs to individual virtual machines (VMs), rather than allowing a single GPU to be shared across multiple VMs.

10.4 CPU support

ROCm requires CPUs that support PCIe™ atomics. Modern CPUs after the release of 1st generation AMD Zen CPU and Intel™ Haswell support PCIe atomics.

[2] Oracle Linux 8 and 9 are supported only on AMD Instinct MI300X.
[3] Azure Linux 3.0 is supported only on AMD Instinct MI300X and AMD Radeon PRO V710.
[4] Debian 12 is supported only on AMD Instinct MI300X for single-node functionality.

CHAPTER

ELEVEN

INSTALLATION TROUBLESHOOTING

Troubleshooting describes issues that some users encounter when installing the ROCm tools or libraries.

11.1 Issue #1: Installation methods

As an example, the latest version of ROCm is 6.0.2, but the installation instructions result in release 6.0.0 being installed.

Solution: You may have used the quick-start installation method which only installs the latest major release. Use one of the other available installation methods:

- Quick-start installation - Installs only the latest major release (i.e. 6.0.0, or 6.1.0)
- Native package manager install method - Installs the specified major and minor release version (i.e. 6.0.0, 6.0.2)
- amdgpu-install method - Installs the specified major and minor release version (i.e. 6.0.0, 6.0.2)

Refer to ROCm Issue #2422 for additional details.

11.2 Issue #2: Install prerequisites

When installing, I see the following message: Problem: nothing provides perl-URI-Encode needed to be installed by ...

Solution: Ensure that the Installation prerequisites are installed. There are prerequisite PERL packages required for SUSE. RHEL also requires Extra Packages for Enterprise Linux (EPEL) to be installed, which is also mentioned in prerequisites. Be sure to install those first, then repeat your installation steps.

Refer to ROCm Issue #1827.

11.3 Issue #3: PATH variable

After successfully installing ROCm, when I run rocminfo (or another ROCm tool) the command is not found.

Solution: You may need to update your PATH environment variable as described in Post-installation instructions.

Refer to ROCm Issue #1607.

11.4 Issue #4: C++ libraries

When compiling HIP programs, I get a linking error for -lstdc++, or fatal error: 'cmath' file not found.

Solution: You can install C++ libraries using your package manager. The following is an Ubuntu example:

```
sudo apt-get install libstdc++-12-dev
```

Refer to ROCm Issue #2031.

11.5 Issue #5: Application hangs on Multi-GPU systems

Running on a system with multiple GPUs the application hangs with the GPU use at 100%, but without the expected GPU temperature buildup

This issue often results in the following message in the application transcript:

```
NCCL WARN Missing "iommu=pt" from kernel command line which can lead to system instablity or␣
↪hang!
```

Solution: To resolve this issue add iommu=pt to GRUB_CMDLINE_LINUX_DEFAULT in /etc/default/grub. Then run the following command:

```
sudo update-grub
```

Reboot the system, and run the following command:

```
cat /proc/cmdline
```

The returned information should reflect the addition of iommu:

```
BOOT_IMAGE=/vmlinuz-5.15.0-101-generic root=/dev/mapper/ubuntu--vg-ubuntu--lv ro iommu=pt
```

Refer to RCCL Issue #1129 for more information.

11.6 Issue #6: Additional packages for Docker installations

Docker images often come with minimal installations, meaning some essential packages might be missing. When installing ROCm within a Docker container, you might need to install additional packages for a successful ROCm installation. Use the following commands to install the prerequisite packages.

Ubuntu

```
apt update
apt install sudo wget
```

RHEL

```
dnf install sudo wget
subscription-manager register --username <username> --password <password>
subscription-manager attach --auto
subscription-manager repos --enable codeready-builder-for-rhel-9-x86_64-rpms
```

SLES

```
zypper install sudo wget SUSEConnect
SUSEConnect -r <REGCODE>
SUSEConnect -p sle-module-desktop-applications/15.5/x86_64
SUSEConnect -p sle-module-development-tools/15.5/x86_64
SUSEConnect -p PackageHub/15.5/x86_64
```

After installing these packages and registering using your license for Enterprise Linux (if applicable), install ROCm following the Quick start installation guide in your Docker container.

11.7 Issue #7: Installations using Python wheels (.whl files) do not support soft links

If you have installed ROCm or any ROCm component using a Python wheel (.whl file), running a ROCm command which is soft-linked will fail with not found on Ubuntu, bad interpreter: No such file or directory on SLES, and ModuleNotFoundError on RHEL.

Solution: Python wheel files do not support soft links (symbolic links). You will need to run soft-linked commands from within their installation directories, or using the full path to their locations.

For example, run rocm-smi on ROCm 6.2 in the following way:

```
cd /opt/rocm-6.2.0/libexec/rocm_smi/
python3 rocm_smi.py
```

or

```
python3 /opt/rocm-6.2.0/libexec/rocm_smi/rocm_smi.py
```

See Symbolic links in wheels for more information.

11.8 Issue #8: The AMDGPU driver is not loaded after installation

When you are verifying the ROCm installation according to the post-install instructions, the rocm-smi and rocminfo commands might fail with the error message Driver not initialized or not display any output. This could indicate the AMDGPU driver is not loaded.

Solution: Ensure the AMDGPU driver is not on a denylist such as /etc/modprobe.d/blacklist-amdgpu.conf. The location of this file might vary depending on the system distribution and version. To verify whether the driver is on a denylist, use the following command:

```
grep amdgpu /etc/modprobe.d/*
```

> **Note**
>
> When installing the AMDGPU driver with Secure Boot enabled, you must sign amdgpu-dkms to prevent potential system loading issues. For more information, see Secure Boot Support. If you prefer not to sign the AMDGPU driver, you can disable Secure Boot from the BIOS settings instead.

11.9 Issue #9: Cannot access the AMD GPU or accelerator after installation

If the group permissions are not set properly during ROCm installation, you might get an error similar to Permission denied when attempting to access the AMD GPU.

Solution: You must be part of the video and render groups to access the AMD GPU or accelerator. To learn how to add an account to these groups, see Configuring permissions for GPU access.

CHAPTER

TWELVE

USER AND KERNEL-SPACE SUPPORT MATRIX

Starting from ROCm™ 6.4.0, forward and backward compatibility between the AMD Kernel-mode GPU Driver (KMD) and its user space software is provided up to a year apart (assuming hardware support is available in both). For earlier ROCm releases, the compatibility is provided for +/- 2 releases. This table shows the compatibility combinations that are currently supported.

> **Note**
>
> The tested user space versions in the following table are accurate as of the time of publication. For the most up-to-date information about AMD Kernel-mode GPU Driver (KMD) and tested user space versions, see the latest version of this table at User and kernel-space support matrix.

KMD	Tested user space versions
6.4.x	6.1.x, 6.2.x, 6.3.x, 6.4.x
6.3.x	6.1.x, 6.2.x, 6.3.x, 6.4.x
6.2.x	6.0.x, 6.1.x, 6.2.x, 6.3.x, 6.4.x
6.1.x	5.7.x, 6.0.x, 6.1.x, 6.2.x, 6.3.x, 6.4.x
6.0.x	5.6.x, 5.7.x, 6.0.x, 6.1.x, 6.2.x
5.7.x	5.5.x, 5.6.x, 5.7.x, 6.0.x, 6.1.x
5.6.x	5.4.x, 5.5.x, 5.6.x, 5.7.x, 6.0.x
5.5.x	5.3.x, 5.4.x, 5.5.x, 5.6.x, 5.7.x
5.4.x	5.2.x, 5.3.x, 5.4.x, 5.5.x, 5.6.x
5.3.x	5.1.x, 5.2.x, 5.3.x, 5.4.x, 5.5.x

> **Note**
>
> For Azure Linux, the latest amdgpu version supported is from the ROCm 6.2.2 release.