ROCm installation on Linux

Release 7.2.3

Advanced Micro Devices, Inc.

May 06, 2026

INSTALL ROCM

This section describes the ROCm for Linux installation options.

> **Note**
>
> - If you're using ROCm with AMD Radeon™ GPUs or Ryzen™ APUs for graphics workloads, see the Use ROCm on Radeon and Ryzen documentation for installation instructions.
> - The AMDGPU installer documentation has been removed to encourage the use of the package manager for ROCm installation. While the package manager is the recommended method, you can still install ROCm using the AMDGPU installer by following the legacy process. Ensure to update the command with the intended ROCm version before running it.

Install ROCm

- System requirements (Linux)
- User and AMD GPU Driver (amdgpu) support matrix
- Quick start - recommended for new users
- Detailed install - includes explanations

Install deep learning frameworks

- PyTorch
- TensorFlow
- JAX
- DGL

How to

- Run Docker containers
- Use Spack

Reference

- Package details
- Troubleshooting

CHAPTER

ONE

SYSTEM REQUIREMENTS (LINUX)

1.1 Supported GPUs

The following table shows the supported AMD Instinct™ GPUs, and Radeon™ PRO and Radeon GPUs. If a GPU is not listed on this table, it's not officially supported by AMD.

GPUs listed in the following table support compute workloads (no display information or graphics). If you're using ROCm with AMD Radeon™ GPUs or Ryzen™ APUs for graphics workloads, see the Use ROCm on Radeon and Ryzen documentation to verify compatibility and system requirements.

> **Note**
>
> If your GPU is not listed, it might be community-enabled through TheRock nightly builds. For more information, see TheRock supported GPUs. For installation guidance, see TheRock releases.

AMD Instinct

GPU	Series	Architecture	LLVM target	Support
AMD Instinct MI355X	MI350	CDNA4	gfx950	[1]
AMD Instinct MI350X	MI350	CDNA4	gfx950	[1]
AMD Instinct MI325X	MI300	CDNA3	gfx942	[2]
AMD Instinct MI300X	MI300	CDNA3	gfx942	[3]
AMD Instinct MI300A	MI300	CDNA3	gfx942	[4]
AMD Instinct MI250X	MI200	CDNA2	gfx90a	[5]
AMD Instinct MI250	MI200	CDNA2	gfx90a	[5]
AMD Instinct MI210	MI200	CDNA2	gfx90a	[5]
AMD Instinct MI100	MI100	CDNA	gfx908	[6]
AMD Instinct MI50	N/A	GCN5.1	gfx906	
AMD Instinct MI25	N/A	GCN5.0	gfx900	

[1] AMD Instinct MI355X and MI350X GPUs supports all Supported operating systems listed below except RHEL 8.10, Debian 12, Rocky Linux 9, and Oracle Linux 8.

[2] AMD Instinct MI325X GPU supports all Supported operating systems listed below except RHEL 8.10, Rocky Linux 9, and Oracle Linux 8.

[3] AMD Instinct MI300X GPU supports all Supported operating systems listed below.

[4] AMD Instinct MI300A GPU supports all Supported operating systems listed below except Debian 13, Oracle Linux 10, Oracle Linux 9, and Oracle Linux 8.

[5] AMD Instinct MI200 Series GPUs support all Supported operating systems listed below except Debian 13, Rocky Linux 9, Oracle Linux 10, Oracle Linux 9, and Oracle Linux 8.

[6] AMD Instinct MI100 GPU supports all Supported operating systems listed below except Debian 13, Debian 12, Rocky Linux 9, Oracle Linux 10, Oracle Linux 9, and Oracle Linux 8.

AMD Radeon PRO

GPU	Architecture	LLVM target	Support
AMD Radeon AI PRO R9700	RDNA4	gfx1201	[7]
AMD Radeon AI PRO R9600D	RDNA4	gfx1201	[7]
AMD Radeon PRO V710	RDNA3	gfx1101	[7]
AMD Radeon PRO W7900 Dual Slot	RDNA3	gfx1100	[7]
AMD Radeon PRO W7900	RDNA3	gfx1100	[7]
AMD Radeon PRO W7800 48GB	RDNA3	gfx1100	[7]
AMD Radeon PRO W7800	RDNA3	gfx1100	[7]
AMD Radeon PRO W7700	RDNA3	gfx1101	[7]
AMD Radeon PRO W6800	RDNA2	gfx1030	[7]
AMD Radeon PRO V620	RDNA2	gfx1030	[8]
AMD Radeon PRO VII	GCN5.1	gfx906	

AMD Radeon

GPU	Architecture	LLVM target	Support
AMD Radeon RX 9070 XT	RDNA4	gfx1201	[7]
AMD Radeon RX 9070 GRE	RDNA4	gfx1201	[7]
AMD Radeon RX 9070	RDNA4	gfx1201	[7]
AMD Radeon RX 9060 XT LP	RDNA4	gfx1200	[7]
AMD Radeon RX 9060 XT	RDNA4	gfx1200	[7]
AMD Radeon RX 9060	RDNA4	gfx1200	[7]
AMD Radeon RX 7900 XTX	RDNA3	gfx1100	[7]
AMD Radeon RX 7900 XT	RDNA3	gfx1100	[7]
AMD Radeon RX 7900 GRE	RDNA3	gfx1100	[7]
AMD Radeon RX 7800 XT	RDNA3	gfx1101	[7]
AMD Radeon RX 7700 XT	RDNA3	gfx1101	[7]
AMD Radeon RX 7700	RDNA3	gfx1101	[7]
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

[7] AMD Radeon PRO (AI PRO R9700, AI PRO R9600D, PRO V710, PRO W7900 Dual Slot, PRO W7900, PRO W7800 48GB, PRO W7800, PRO W7700, and PRO W6800) and AMD Radeon (RX 9070 XT, RX 9070 GRE, RX 9070, RX 9060 XT LP, RX 9060 XT, RX 9060, RX 7900 XTX, RX 7900 XT, RX 7900 GRE, RX 7800 XT, RX 7700 XT, and RX 7700) only support Ubuntu 24.04.4, Ubuntu 22.04.5, RHEL 10.1, and RHEL 9.7.

[8] AMD Radeon PRO V620 only supports Ubuntu 24.04.4 and Ubuntu 22.04.5.

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.

1.2 Supported operating systems

AMD ROCm software supports the following Linux distributions.

Operating system	Kernel	Glibc	Support
Ubuntu 24.04.4	6.8 [GA], 6.17 [HWE]	2.39	
Ubuntu 22.04.5	5.15 [GA], 6.8 [HWE]	2.35	
RHEL 10.1	6.12.0-124	2.39	[9]
RHEL 10.0	6.12.0-55	2.39	[9]
RHEL 9.7	5.14.0-611	2.34	[9]
RHEL 9.6	5.14.0-570	2.34	[9]
RHEL 9.4	5.14.0-427	2.34	[10]
RHEL 8.10	4.18.0-553	2.28	[11]
SLES 15 SP7	6.4.0-150700.51	2.38	[12]
Debian 13	6.12	2.35	[13]
Debian 12	6.1.0	2.36	[14]
Rocky Linux 9	5.14.0-570	2.34	[15]
Oracle Linux 10	6.12.0 (UEK)	2.39	[16]
Oracle Linux 9	5.15.0 (UEK)	2.34	[16]
Oracle Linux 8	5.15.0 (UEK)	2.28	[17]

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.

[9] RHEL 10.1 and RHEL 9.7 are supported on all listed Supported GPUs except AMD Radeon PRO V620 GPU.
[10] RHEL 10.0, RHEL 9.6, and RHEL 9.4 are supported on all AMD Instinct GPUs listed under Supported GPUs.
[11] RHEL 8.10 is supported only on AMD Instinct MI300X, MI300A, MI250X, MI250, MI210, and MI100 GPUs.
[12] SLES 15 SP7 is supported on all AMD Instinct GPUs listed under Supported GPUs.
[13] Debian 13 is supported only on AMD Instinct MI355X, MI350X, MI325X, and MI300X GPUs.
[14] Debian 12 are supported only on AMD Instinct MI325X, MI300X, MI300A, MI250X, MI250, and MI210 GPUs.
[15] Rocky Linux 9 is supported only on AMD Instinct MI300X and MI300A GPUs.
[16] Oracle Linux 10 and 9 are supported only on AMD Instinct MI355X, MI350X, MI325X, and MI300X GPUs.
[17] Oracle Linux 8 is supported only on AMD Instinct MI300X GPUs.

1.3 Virtualization support

ROCm supports virtualization for the Instinct GPUs and Radeon PRO GPUs listed in the following table.

Important

GPU virtualization with KVM-based SR-IOV requires AMD GPU Virtualization Driver (GIM) driver. Refer to GIM Release note.

Note

AMD Virtualization supports the following:

- Passthrough: All 8 GPUs are assigned directly to a single virtual machine (VM).
- SR-IOV: Provides 1 Virtual Function (VF) per GPU (8 VFs in total), which can be flexibly assigned among multiple VMs (for example, 8 VMs with 1 VF each, 4 VMs with 2 VFs each, or 2 VMs with 4 VFs each)

1.4 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.

CHAPTER

TWO

USER AND AMD GPU DRIVER (AMDGPU) SUPPORT MATRIX

The AMD GPU Driver (amdgpu) is distributed separately from the ROCm software stack and is stored under in its own location /amdgpu/ in the package repository at repo.radeon.com. Starting from ROCm 6.4.0, forward and backward compatibility between the AMD GPU Driver (amdgpu) 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 supported user space versions in the following table are accurate as of the time of publication. For the most up-to-date information about AMD GPU Driver (amdgpu) and supported user space versions, see the latest version of this table at User and AMD GPU Driver (amdgpu) support matrix.

AMD GPU Driver (amdgpu)	Supported user space versions
30.30.x	6.3.x, 6.4.x, 7.0.x, 7.1.x, 7.2.x
30.20.x	6.3.x, 6.4.x, 7.0.x, 7.1.x, 7.2.x
30.10.x	6.2.x, 6.3.x, 6.4.x, 7.0.x, 7.1.x, 7.2.x
6.4.x	6.1.x, 6.2.x, 6.3.x, 6.4.x, 7.0.x, 7.1.x, 7.2.x
6.3.x	6.1.x, 6.2.x, 6.3.x, 6.4.x, 7.0.x
6.2.x	6.0.x, 6.1.x, 6.2.x, 6.3.x, 6.4.x, 7.0.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

CHAPTER

THREE

QUICK START INSTALLATION GUIDE

> **Note**
>
> See Use ROCm on Radeon and Ryzen for instructions on installing ROCm on systems with AMD Radeon™ GPUs or Ryzen™ APUs for graphics workloads.

Before proceeding, ensure your kernel meets the ROCm system requirements. Then select your operating system and version, and run the provided commands to install the AMD GPU and ROCm.

For detailed guidance, see Installation via native package manager for AMD GPU installation and Detailed install for ROCm installation.

3.1 Installing

3.1.1 Register repositories

Ubuntu

24.04

```
wget https://repo.radeon.com/amdgpu-install/7.2.3/ubuntu/noble/amdgpu-install_7.2.3.70203-1_all.deb
sudo apt install ./amdgpu-install_7.2.3.70203-1_all.deb
sudo apt update
```

22.04

```
wget https://repo.radeon.com/amdgpu-install/7.2.3/ubuntu/jammy/amdgpu-install_7.2.3.70203-1_all.deb
sudo apt install ./amdgpu-install_7.2.3.70203-1_all.deb
sudo apt update
```

Debian

13

```
wget https://repo.radeon.com/amdgpu-install/7.2.3/ubuntu/noble/amdgpu-install_7.2.3.70203-1_all.deb
sudo apt install ./amdgpu-install_7.2.3.70203-1_all.deb
sudo apt update
```

12

```
wget https://repo.radeon.com/amdgpu-install/7.2.3/ubuntu/jammy/amdgpu-install_7.2.3.70203-1_all.deb
sudo apt install ./amdgpu-install_7.2.3.70203-1_all.deb
sudo apt update
```

Red Hat Enterprise Linux

10.1

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

```
sudo dnf install https://repo.radeon.com/amdgpu-install/7.2.3/rhel/10/amdgpu-install-7.2.3.70203-1.el10.
↪noarch.rpm
sudo dnf clean all
```

10.0

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

```
sudo dnf install https://repo.radeon.com/amdgpu-install/7.2.3/rhel/10/amdgpu-install-7.2.3.70203-1.el10.
↪noarch.rpm
sudo dnf clean all
```

9.7

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

```
sudo dnf install https://repo.radeon.com/amdgpu-install/7.2.3/rhel/9.7/amdgpu-install-7.2.3.70203-1.el9.
↪noarch.rpm
sudo dnf clean all
```

9.6

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

```
sudo dnf install https://repo.radeon.com/amdgpu-install/7.2.3/rhel/9.6/amdgpu-install-7.2.3.70203-1.el9.
↪noarch.rpm
sudo dnf clean all
```

9.4

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

```
sudo dnf install https://repo.radeon.com/amdgpu-install/7.2.3/rhel/9.4/amdgpu-install-7.2.3.70203-1.el9.
↪noarch.rpm
sudo dnf clean all
```

8.10

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

```
sudo dnf install https://repo.radeon.com/amdgpu-install/7.2.3/rhel/8/amdgpu-install-7.2.3.70203-1.el8.
↪noarch.rpm
sudo dnf clean all
```

Oracle Linux

10.1

Before installing ROCm on OL, update your Enterprise Linux.

```
sudo dnf install https://repo.radeon.com/amdgpu-install/7.2.3/el/10/amdgpu-install-7.2.3.70203-1.el10.
↪noarch.rpm
sudo dnf clean all
```

9.7

Before installing ROCm on OL, update your Enterprise Linux.

```
sudo dnf install https://repo.radeon.com/amdgpu-install/7.2.3/el/9.7/amdgpu-install-7.2.3.70203-1.el9.
↪noarch.rpm
sudo dnf clean all
```

8.10

Before installing ROCm on OL, update your Enterprise Linux.

```
sudo dnf install https://repo.radeon.com/amdgpu-install/7.2.3/el/8/amdgpu-install-7.2.3.70203-1.el8.
↪noarch.rpm
sudo dnf clean all
```

SUSE Linux Enterprise Server

15.7

Before installing ROCm on SLES, register and update your Enterprise Linux.

```
sudo SUSEConnect -p sle-module-desktop-applications/15.7/x86_64
sudo SUSEConnect -p sle-module-development-tools/15.7/x86_64
sudo SUSEConnect -p PackageHub/15.7/x86_64
sudo zypper install zypper
sudo zypper --no-gpg-checks install https://repo.radeon.com/amdgpu-install/7.2.3/sle/15.7/amdgpu-
↪install-7.2.3.70203-1.noarch.rpm
sudo zypper --gpg-auto-import-keys refresh
```

Rocky Linux

9.7

```
sudo dnf install https://repo.radeon.com/amdgpu-install/7.2.3/el/9.7/amdgpu-install-7.2.3.70203-1.el9.
↪noarch.rpm
sudo dnf clean all
```

3.1.2 Install kernel driver

Ubuntu

24.04

> **Caution**
>
> Remove any AMD GPU driver from a previous installation by following the uninstall steps in Uninstall kernel driver.

```
sudo apt install "linux-headers-$(uname -r)" "linux-modules-extra-$(uname -r)"
sudo apt install amdgpu-dkms
```

22.04

> **Caution**
>
> Remove any AMD GPU driver from a previous installation by following the uninstall steps in Uninstall kernel driver.

```
sudo apt install "linux-headers-$(uname -r)" "linux-modules-extra-$(uname -r)"
sudo apt install amdgpu-dkms
```

Debian

13

> **Caution**
>
> Remove any AMD GPU driver from a previous installation by following uninstallation steps in Uninstall kernel driver.

```
sudo apt install "linux-headers-$(uname -r)"
sudo apt install amdgpu-dkms
```

12

> **Caution**
>
> Remove any AMD GPU driver from a previous installation by following uninstallation steps in Uninstall kernel driver.

```
sudo apt install "linux-headers-$(uname -r)"
sudo apt install amdgpu-dkms
```

Red Hat Enterprise Linux

10.1

> **Caution**
>
> Remove any AMD GPU driver from a previous installation by following uninstallation steps in Uninstall kernel driver.

```
sudo dnf install "kernel-headers-$(uname -r)" "kernel-devel-$(uname -r)" "kernel-devel-matched-$(uname -r)"
sudo dnf install amdgpu-dkms
```

10.0

> **Caution**
>
> Remove any AMD GPU driver from a previous installation by following uninstallation steps in Uninstall kernel driver.

```
sudo dnf install "kernel-headers-$(uname -r)" "kernel-devel-$(uname -r)" "kernel-devel-matched-$(uname -r)"
sudo dnf install amdgpu-dkms
```

9.7

> **Caution**
>
> Remove any AMD GPU driver from a previous installation by following uninstallation steps in Uninstall kernel driver.

```
sudo dnf install "kernel-headers-$(uname -r)" "kernel-devel-$(uname -r)" "kernel-devel-matched-$(uname -r)"
sudo dnf install amdgpu-dkms
```

9.6

> **Caution**
>
> Remove any AMD GPU driver from a previous installation by following uninstallation steps in Uninstall kernel driver.

```
sudo dnf install "kernel-headers-$(uname -r)" "kernel-devel-$(uname -r)" "kernel-devel-matched-$(uname -r)"
sudo dnf install amdgpu-dkms
```

9.4

> **Caution**
>
> Remove any AMD GPU driver from a previous installation by following uninstallation steps in Uninstall kernel driver.

```
sudo dnf install "kernel-headers-$(uname -r)" "kernel-devel-$(uname -r)" "kernel-devel-matched-$(uname -r)"
sudo dnf install amdgpu-dkms
```

8.10

> **Caution**
>
> Remove any AMD GPU driver from a previous installation by following uninstallation steps in Uninstall kernel driver.

```
sudo dnf install "kernel-headers-$(uname -r)" "kernel-devel-$(uname -r)"
sudo dnf install amdgpu-dkms
```

Oracle Linux

10.1

> **Caution**
>
> Remove any AMD GPU driver from a previous installation by following uninstallation steps in Uninstall kernel driver.

```
sudo dnf install "kernel-uek-devel-$(uname -r)"
sudo dnf install amdgpu-dkms
```

9.7

> **Caution**
>
> Remove any AMD GPU driver from a previous installation by following uninstallation steps in Uninstall kernel driver.

```
sudo dnf install "kernel-uek-devel-$(uname -r)"
sudo dnf install amdgpu-dkms
```

8.10

> **Caution**
>
> Remove any AMD GPU driver from a previous installation by following uninstallation steps in Uninstall kernel driver.

```
sudo dnf install "kernel-uek-devel-$(uname -r)"
sudo dnf install amdgpu-dkms
```

SUSE Linux Enterprise Server

15.7

> **Caution**
>
> Remove any AMD GPU driver from a previous installation by following uninstallation steps in Uninstall kernel driver.

```
sudo zypper install kernel-default-devel
sudo zypper install amdgpu-dkms
```

Rocky Linux

9.7

> **Caution**
>
> Remove any AMD GPU driver from a previous installation by following uninstallation steps in Uninstall kernel driver.

```
sudo dnf install "kernel-headers" "kernel-devel" "kernel-devel-matched"
sudo dnf install amdgpu-dkms
```

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

3.1.3 Install ROCm

Ubuntu

24.04

```
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

```
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

13

```
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
```

12

```
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
```

Red Hat Enterprise Linux

10.1

```
wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-10.noarch.rpm
sudo rpm -ivh epel-release-latest-10.noarch.rpm
sudo dnf config-manager --enable codeready-builder-for-rhel-10-x86_64-rpms
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
```

10.0

```
wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-10.noarch.rpm
sudo rpm -ivh epel-release-latest-10.noarch.rpm
sudo dnf config-manager --enable codeready-builder-for-rhel-10-x86_64-rpms
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
```

9.7

```
wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.rpm
sudo rpm -ivh epel-release-latest-9.noarch.rpm
sudo dnf config-manager --enable codeready-builder-for-rhel-9-x86_64-rpms
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
```

9.6

```
wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.rpm
sudo rpm -ivh epel-release-latest-9.noarch.rpm
sudo dnf config-manager --enable codeready-builder-for-rhel-9-x86_64-rpms
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

```
wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.rpm
sudo rpm -ivh epel-release-latest-9.noarch.rpm
sudo dnf config-manager --enable codeready-builder-for-rhel-9-x86_64-rpms
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

```
wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
sudo rpm -ivh epel-release-latest-8.noarch.rpm
sudo dnf config-manager --enable codeready-builder-for-rhel-8-x86_64-rpms
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

10.1

```
wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-10.noarch.rpm
sudo rpm -ivh epel-release-latest-10.noarch.rpm
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
```

9.7

```
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
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

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

(continues on next page)

(continued from previous page)

```
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

15.7

```
sudo zypper addrepo https://download.opensuse.org/repositories/science/SLE_15_SP5/science.repo
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
```

Rocky Linux

9.7

```
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
```

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.

3.2 Uninstalling

3.2.1 Uninstall ROCm

Ubuntu

24.04

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

22.04

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

Debian

13

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

12

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

Red Hat Enterprise Linux

10.1

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

10.0

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

9.7

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

9.6

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

9.4

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

8.10

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

Oracle Linux

10.1

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

9.7

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

8.10

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

SUSE Linux Enterprise Server

15.7

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

Rocky Linux

9.7

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

3.2.2 Uninstall kernel driver

Ubuntu

24.04

```
sudo apt autoremove amdgpu-dkms
```

22.04

```
sudo apt autoremove amdgpu-dkms
```

Debian

13

```
sudo apt autoremove amdgpu-dkms
```

12

```
sudo apt autoremove amdgpu-dkms
```

Red Hat Enterprise Linux

10.1

```
sudo dnf remove amdgpu-dkms
```

10.0

```
sudo dnf remove amdgpu-dkms
```

9.7

```
sudo dnf remove amdgpu-dkms
```

9.6

```
sudo dnf remove amdgpu-dkms
```

9.4

```
sudo dnf remove amdgpu-dkms
```

8.10

```
sudo dnf remove amdgpu-dkms
```

Oracle Linux

10.1

```
sudo dnf remove amdgpu-dkms
```

9.7

```
sudo dnf remove amdgpu-dkms
```

8.10

```
sudo dnf remove amdgpu-dkms
```

SUSE Linux Enterprise Server

15.7

```
sudo zypper remove amdgpu-dkms amdgpu-dkms-firmware
```

Rocky Linux

9.7

```
sudo dnf remove amdgpu-dkms
```

Important

To apply all settings, reboot your system.

3.2.3 Remove repositories

Ubuntu

24.04

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

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

22.04

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

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

Debian

13

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

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

12

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

# Clear the cache and clean the system
```

(continues on next page)

(continued from previous page)

```
sudo rm -rf /var/cache/apt/*
sudo apt clean all
sudo apt update
```

Red Hat Enterprise Linux

10.1

```
sudo dnf remove amdgpu-install

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

10.0

```
sudo dnf remove amdgpu-install

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

9.7

```
sudo dnf remove amdgpu-install

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

9.6

```
sudo dnf remove amdgpu-install

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

9.4

```
sudo dnf remove amdgpu-install

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

8.10

```
sudo dnf remove amdgpu-install

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

Oracle Linux

10.1

```
sudo dnf remove amdgpu-install

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

9.7

```
sudo dnf remove amdgpu-install

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

8.10

```
sudo dnf remove amdgpu-install

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

SUSE Linux Enterprise Server

15.7

```
sudo zypper remove amdgpu-install

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

Rocky Linux

9.7

```
sudo dnf remove amdgpu-install

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

CHAPTER

FOUR

DETAILED INSTALL

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.

4.1 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=24.04
     DISTRIB_CODENAME=noble
     DISTRIB_DESCRIPTION="Ubuntu 24.04.3 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 6.8.0-50-generic #51-Ubuntu SMP PREEMPT_DYNAMIC Sat Nov  9 17:58:29 UTC␣
     ↪2024 x86_64
     ```

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

4.1.1 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

10.1

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>
```

More details about registering for RHEL

10.0

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>
```

More details about registering for RHEL

9.7

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

9.6

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

9.4

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

8.10

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

Rocky Linux

There is no registration required for Rocky Linux.

4.1.2 Update your Enterprise Linux

If you are using Red Hat Enterprise Linux (RHEL), SUSE Linux Enterprise Servers (SLES), or Oracle Linux (OL), 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

10.1

```
sudo dnf update redhat-release
sudo dnf update --releasever=10.1 --exclude=\*release\*
```

10.0

```
sudo dnf update --releasever=10.0 --exclude=\*release\*
```

9.7

```
sudo dnf update --releasever=9.7 --exclude=\*release\*
```

9.6

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

9.4

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

8.10

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

Oracle Linux

10.1

```
sudo dnf update --releasever=10.1 --exclude=\*release\*
```

9.7

```
sudo dnf update --releasever=9.7 --exclude=\*release\*
```

8.10

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

SUSE Linux Enterprise Server

```
sudo zypper update
```

Rocky Linux

There is no update required for Rocky Linux.

Important

To apply all settings, reboot your system.

4.1.3 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.

   ##### 10.1

   ```
   wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-10.noarch.rpm
   sudo rpm -ivh epel-release-latest-10.noarch.rpm
   ```

   ##### 10.0

   ```
   wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-10.noarch.rpm
   sudo rpm -ivh epel-release-latest-10.noarch.rpm
   ```

   ##### 9.7

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

   ##### 9.6

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

   ##### 9.4

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

   ##### 8.10

   ```
   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.

10.1

```
sudo dnf config-manager --enable codeready-builder-for-rhel-10-x86_64-rpms
```

10.0

```
sudo dnf config-manager --enable codeready-builder-for-rhel-10-x86_64-rpms
```

9.7

```
sudo dnf config-manager --enable codeready-builder-for-rhel-9-x86_64-rpms
```

9.6

```
sudo dnf config-manager --enable codeready-builder-for-rhel-9-x86_64-rpms
```

9.4

```
sudo dnf config-manager --enable codeready-builder-for-rhel-9-x86_64-rpms
```

8.10

```
sudo dnf config-manager --enable codeready-builder-for-rhel-8-x86_64-rpms
```

Oracle Linux

1. Add the EPEL repository.

10.1

```
wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-10.noarch.rpm
sudo rpm -ivh epel-release-latest-10.noarch.rpm
```

9.7

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

8.10

```
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.

```
sudo crb enable
```

SUSE Linux Enterprise Server

Add a few modules with SUSEConnect and the science repository.

15.7

```
sudo SUSEConnect -p sle-module-desktop-applications/15.7/x86_64
sudo SUSEConnect -p sle-module-development-tools/15.7/x86_64
sudo SUSEConnect -p PackageHub/15.7/x86_64
sudo zypper install zypper
sudo zypper addrepo https://download.opensuse.org/repositories/science/SLE_15_SP5/science.repo
```

Rocky Linux

1. Add the EPEL repository.

   ##### 9.7

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

4.1.4 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
```

Rocky Linux

```
sudo dnf 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
```

Rocky Linux

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

4.1.5 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.

4.1.5.1 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
```

4.1.5.2 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.

GPU access may be granted to either all users or a custom group:

4.1.5.2.1 a. Grant GPU access to all users on the system

To set up udev rules, install the package using the following instructions specific to your distribution:

Ubuntu

24.04

```
sudo apt update
wget https://repo.radeon.com/amdgpu/30.30.3/ubuntu/pool/main/a/amdgpu-insecure-instinct-udev-
↪rules/amdgpu-insecure-instinct-udev-rules_30.30.3.0-2327507.24.04_all.deb
sudo apt install ./amdgpu-insecure-instinct-udev-rules_30.30.3.0-2327507.24.04_all.deb
```

22.04

```
sudo apt update
wget https://repo.radeon.com/amdgpu/30.30.3/ubuntu/pool/main/a/amdgpu-insecure-instinct-udev-
↪rules/amdgpu-insecure-instinct-udev-rules_30.30.3.0-2327507.22.04_all.deb
sudo apt install ./amdgpu-insecure-instinct-udev-rules_30.30.3.0-2327507.22.04_all.deb
```

Debian

13

```
sudo apt update
wget https://repo.radeon.com/amdgpu/30.30.3/ubuntu/pool/main/a/amdgpu-insecure-instinct-udev-
↪rules/amdgpu-insecure-instinct-udev-rules_30.30.3.0-2327507.24.04_all.deb
sudo apt install ./amdgpu-insecure-instinct-udev-rules_30.30.3.0-2327507.24.04_all.deb
```

12

```
sudo apt update
wget https://repo.radeon.com/amdgpu/30.30.3/ubuntu/pool/main/a/amdgpu-insecure-instinct-udev-
↪rules/amdgpu-insecure-instinct-udev-rules_30.30.3.0-2327507.22.04_all.deb
sudo apt install ./amdgpu-insecure-instinct-udev-rules_30.30.3.0-2327507.22.04_all.deb
```

Red Hat Enterprise Linux

10.1

```
sudo dnf install https://repo.radeon.com/amdgpu/30.30.3/el/10/main/x86_64/amdgpu-insecure-instinct-
↪udev-rules-30.30.3.0-2327507.el10.noarch.rpm
```

10.0

```
sudo dnf install https://repo.radeon.com/amdgpu/30.30.3/el/10/main/x86_64/amdgpu-insecure-instinct-
↪udev-rules-30.30.3.0-2327507.el10.noarch.rpm
```

9.7

```
sudo dnf install https://repo.radeon.com/amdgpu/30.30.3/el/9.7/main/x86_64/amdgpu-insecure-instinct-
↪udev-rules-30.30.3.0-2327507.el9.noarch.rpm
```

9.6

```
sudo dnf install https://repo.radeon.com/amdgpu/30.30.3/el/9.6/main/x86_64/amdgpu-insecure-instinct-
↪udev-rules-30.30.3.0-2327507.el9.noarch.rpm
```

9.4

```
sudo dnf install https://repo.radeon.com/amdgpu/30.30.3/el/9.4/main/x86_64/amdgpu-insecure-instinct-
↪udev-rules-30.30.3.0-2327507.el9.noarch.rpm
```

8.10

```
sudo dnf install https://repo.radeon.com/amdgpu/30.30.3/el/8/main/x86_64/amdgpu-insecure-instinct-
↪udev-rules-30.30.3.0-2327507.el8.noarch.rpm
```

Oracle Linux

10.1

```
sudo dnf install https://repo.radeon.com/amdgpu/30.30.3/el/10/main/x86_64/amdgpu-insecure-instinct-
↪udev-rules-30.30.3.0-2327507.el10.noarch.rpm
```

9.7

```
sudo dnf install https://repo.radeon.com/amdgpu/30.30.3/el/9.7/main/x86_64/amdgpu-insecure-instinct-
↪udev-rules-30.30.3.0-2327507.el9.noarch.rpm
```

8.10

```
sudo dnf install https://repo.radeon.com/amdgpu/30.30.3/el/8/main/x86_64/amdgpu-insecure-instinct-
↪udev-rules-30.30.3.0-2327507.el8.noarch.rpm
```

SUSE Linux Enterprise Server

15.7

```
sudo zypper --no-gpg-checks install https://repo.radeon.com/amdgpu/30.30.3/sle/15.7/main/x86_64/
↪amdgpu-insecure-instinct-udev-rules-30.30.3.0-2327507.noarch.rpm
```

Rocky Linux

9.7

```
sudo dnf install https://repo.radeon.com/amdgpu/30.30.3/el/9.7/main/x86_64/amdgpu-insecure-instinct-
↪udev-rules-30.30.3.0-2327507.el9.noarch.rpm
```

4.1.5.2.2 b. 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.

4.1.6 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.

4.1.7 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.

4.2 ROCm installation overview

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

> **Note**
>
> - If you're using ROCm with AMD Radeon™ GPUs or Ryzen™ APUs for graphics workloads, see the Use ROCm on Radeon and Ryzen.
> - The AMDGPU installer documentation has been removed to encourage the use of the package manager for ROCm installation. While the package manager is the recommended method, you can still install ROCm using the AMDGPU installer by following the legacy process. Ensure to update the command with the intended ROCm version before running it.

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

4.2.1 Installation methods

- Package manager
- Multi-version installation
- ROCm Runfile Installer

4.2.1.1 Package manager

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. See Installation via native package manager for instructions based on the operating system.

4.2.1.2 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.

4.2.1.3 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.

4.2.2 Installation via native package manager

Select the install instructions for your operating system

Install

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

Uninstall

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

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

4.2.2.1 Ubuntu native installation

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

> **Note**
>
> The following installation steps also apply when upgrading from a previous ROCm version.

4.2.2.1.1 Registering ROCm repositories

4.2.2.1.1.1 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.

4.2.2.1.1.2 Register packages

Ubuntu 24.04

```
sudo tee /etc/apt/sources.list.d/rocm.list << EOF
deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/rocm/apt/7.2.3␣
↪noble main
deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/graphics/7.2.3/
↪ubuntu noble main
EOF

sudo tee /etc/apt/preferences.d/rocm-pin-600 << EOF
Package: *
Pin: release o=repo.radeon.com
Pin-Priority: 600
EOF

sudo apt update
```

Ubuntu 22.04

```
sudo tee /etc/apt/sources.list.d/rocm.list << EOF
deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/rocm/apt/7.2.3␣
↪jammy main
deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/graphics/7.2.3/
↪ubuntu jammy main
EOF

sudo tee /etc/apt/preferences.d/rocm-pin-600 << EOF
Package: *
Pin: release o=repo.radeon.com
Pin-Priority: 600
EOF

sudo apt update
```

4.2.2.1.2 Installing

4.2.2.1.2.1 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 AMD GPU Driver (amdgpu) support matrix.

4.2.2.1.2.2 Install ROCm

```
sudo apt install rocm
```

ROCm installation can be tailored to your requirements using one more combinations of ROCm meta packages:

- To use pre-built ROCm libraries and tools, include ROCm runtime packages in the installation step.
- To develop and build individual ROCm libraries and tools, include ROCm developer packages in the installation step.

4.2.2.1.2.3 ROCm runtime packages

Meta package	Description	Legacy use case[1]
rocm	All ROCm core packages, tools, and libraries.	rocm
rocm-hip-libraries	HIP libraries optimized for the AMD platform.	Legacy use case does not exist.
rocm-hip-runtime	Run HIP applications written for the AMD platform.	hip
rocm-language-runti	ROCm runtime environment for running applications on the AMD platform.	lrt
rocm-ml-libraries	Key machine learning libraries. Includes MIOpen.	mllib
rocm-opencl-runtim	Run OpenCL-based applications on the AMD platform.	opencl
Other package		
amdgpu-lib	For users of graphics applications which require the open source Mesa 3D graphics and multimedia libraries. This package is primarily used for Radeon GPUs.	graphics

4.2.2.1.2.4 ROCm developer packages

Meta package	Description	Legacy use case[1]
rocm-developer-tool	Debug and profile HIP applications.	rocmdevtools
rocm-hip-runtime-d	Develop applications on HIP or port from CUDA.	Legacy use case does not exist.
rocm-hip-runtime-d		
rocm-hip-sdk[3]	Develop or port HIP applications and libraries for the AMD platform.	hiplibsdk
rocm-ml-sdk	Develop and run machine learning applications for AMD.	mlsdk
rocm-opencl-sdk	Develop OpenCL-based applications for the AMD platform.	openclsdk
rocm-openmp-sdk	Develop OpenMP-based applications for the AMD software.	openmpsdk

4.2.2.1.3 Post-installation

Complete the Post-installation instructions.

4.2.2.1.4 Uninstalling

4.2.2.1.4.1 Uninstall ROCm meta packages

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

[1] Starting from ROCm 6.4.2, "Legacy use cases" in amdgpu-install are replaced by the equivalent meta package. In addition, the following amdgpu-install use cases: asan, rocmdev, multimedia, multimediasdk, amf, and workstation are deprecated.

[2] Use rocm-hip-runtime-dev for Debian/Ubuntu-based systems and rocm-hip-runtime-devel for RHEL/RPM-based systems.

[3] rocm-hip-sdk requires rocm-hip-runtime-devel.

4.2.2.1.4.2 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
```

Important

To apply all settings, reboot your system.

4.2.2.2 Debian native installation

Caution

Ensure that the Installation prerequisites are met.

Note

The following installation steps also apply when upgrading from a previous ROCm version.

4.2.2.2.1 Registering ROCm repositories

4.2.2.2.1.1 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.

4.2.2.2.1.2 Register packages

Debian 13

```
# Register ROCm packages
sudo tee /etc/apt/sources.list.d/rocm.list << EOF
deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/rocm/apt/7.2.3␣
↪noble main
deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/graphics/7.2.3/
↪ubuntu noble main
EOF

sudo tee /etc/apt/preferences.d/rocm-pin-600 << EOF
Package: *
Pin: release o=repo.radeon.com
Pin-Priority: 600
EOF

sudo apt update
```

Debian 12

```
# Register ROCm packages
sudo tee /etc/apt/sources.list.d/rocm.list << EOF
deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/rocm/apt/7.2.3␣
↪jammy main
deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/graphics/7.2.3/
↪ubuntu jammy main
EOF

sudo tee /etc/apt/preferences.d/rocm-pin-600 << EOF
Package: *
Pin: release o=repo.radeon.com
Pin-Priority: 600
EOF

sudo apt update
```

4.2.2.2.2 Installing

4.2.2.2.2.1 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 AMD GPU Driver (amdgpu) support matrix.

4.2.2.2.2.2 Install ROCm

```
sudo apt install rocm
```

ROCm installation can be tailored to your requirements using one more combinations of ROCm meta packages:

- To use pre-built ROCm libraries and tools, include ROCm runtime packages in the installation step.
- To develop and build individual ROCm libraries and tools, include ROCm developer packages in the installation step.

4.2.2.2.2.3 ROCm runtime packages

Meta package	Description	Legacy use case[1]
rocm	All ROCm core packages, tools, and libraries.	rocm
rocm-hip-libraries	HIP libraries optimized for the AMD platform.	Legacy use case does not exist.
rocm-hip-runtime	Run HIP applications written for the AMD platform.	hip
rocm-language-runti	ROCm runtime environment for running applications on the AMD platform.	lrt
rocm-ml-libraries	Key machine learning libraries. Includes MIOpen.	mllib
rocm-opencl-runtim	Run OpenCL-based applications on the AMD platform.	opencl
Other package		
amdgpu-lib	For users of graphics applications which require the open source Mesa 3D graphics and multimedia libraries. This package is primarily used for Radeon GPUs.	graphics

4.2.2.2.2.4 ROCm developer packages

Meta package	Description	Legacy use case[1]
rocm-developer-tool	Debug and profile HIP applications.	rocmdevtools
rocm-hip-runtime-de	Develop applications on HIP or port from CUDA.	Legacy use case does not exist.
rocm-hip-runtime-de		
rocm-hip-sdk[3]	Develop or port HIP applications and libraries for the AMD platform.	hiplibsdk
rocm-ml-sdk	Develop and run machine learning applications for AMD.	mlsdk
rocm-opencl-sdk	Develop OpenCL-based applications for the AMD platform.	openclsdk
rocm-openmp-sdk	Develop OpenMP-based applications for the AMD software.	openmpsdk

4.2.2.2.3 Post-installation

Complete the Post-installation instructions.

[1] Starting from ROCm 6.4.2, "Legacy use cases" in amdgpu-install are replaced by the equivalent meta package. In addition, the following amdgpu-install use cases: asan, rocmdev, multimedia, multimediasdk, amf, and workstation are deprecated.
[2] Use rocm-hip-runtime-dev for Debian/Ubuntu-based systems and rocm-hip-runtime-devel for RHEL/RPM-based systems.
[3] rocm-hip-sdk requires rocm-hip-runtime-devel.

4.2.2.2.4 Uninstalling

4.2.2.2.4.1 Uninstall ROCm meta packages

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

4.2.2.2.4.2 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
```

Important

To apply all settings, reboot your system.

4.2.2.3 Red Hat Enterprise Linux native installation

Caution

Ensure that the Installation prerequisites are met.

Note

The following installation steps also apply when upgrading from a previous ROCm version.

4.2.2.3.1 Registering ROCm repositories

RHEL 10.1

```
sudo tee /etc/yum.repos.d/rocm.repo <<EOF
[rocm]
name=ROCm 7.2.3 repository
baseurl=https://repo.radeon.com/rocm/el10/7.2.3/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key

[amdgraphics]
name=AMD Graphics 7.2.3 repository
baseurl=https://repo.radeon.com/graphics/7.2.3/el/10/main/x86_64/
enabled=1
```

(continues on next page)

(continued from previous page)

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

RHEL 10.0

```
sudo tee /etc/yum.repos.d/rocm.repo <<EOF
[rocm]
name=ROCm 7.2.3 repository
baseurl=https://repo.radeon.com/rocm/el10/7.2.3/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key

[amdgraphics]
name=AMD Graphics 7.2.3 repository
baseurl=https://repo.radeon.com/graphics/7.2.3/el/10/main/x86_64/
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
sudo dnf clean all
```

RHEL 9.7

```
sudo tee /etc/yum.repos.d/rocm.repo <<EOF
[rocm]
name=ROCm 7.2.3 repository
baseurl=https://repo.radeon.com/rocm/el9/7.2.3/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key

[amdgraphics]
name=AMD Graphics 7.2.3 repository
baseurl=https://repo.radeon.com/graphics/7.2.3/el/9.7/main/x86_64/
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
sudo dnf clean all
```

RHEL 9.6

```
sudo tee /etc/yum.repos.d/rocm.repo <<EOF
[rocm]
name=ROCm 7.2.3 repository
baseurl=https://repo.radeon.com/rocm/el9/7.2.3/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key

[amdgraphics]
name=AMD Graphics 7.2.3 repository
baseurl=https://repo.radeon.com/graphics/7.2.3/el/9.6/main/x86_64/
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
sudo dnf clean all
```

RHEL 9.4

```
sudo tee /etc/yum.repos.d/rocm.repo <<EOF
[rocm]
name=ROCm 7.2.3 repository
baseurl=https://repo.radeon.com/rocm/el9/7.2.3/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key

[amdgraphics]
name=AMD Graphics 7.2.3 repository
baseurl=https://repo.radeon.com/graphics/7.2.3/el/9.4/main/x86_64/
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
sudo dnf clean all
```

RHEL 8.10

```
sudo tee /etc/yum.repos.d/rocm.repo <<EOF
[rocm]
name=ROCm 7.2.3 repository
baseurl=https://repo.radeon.com/rocm/el8/7.2.3/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
```

(continues on next page)

(continued from previous page)

```
[amdgraphics]
name=AMD Graphics 7.2.3 repository
baseurl=https://repo.radeon.com/graphics/7.2.3/el/8/main/x86_64/
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
sudo dnf clean all
```

4.2.2.3.2 Installing

4.2.2.3.2.1 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 AMD GPU Driver (amdgpu) support matrix.

4.2.2.3.2.2 Install ROCm

```
sudo dnf install rocm
```

ROCm installation can be tailored to your requirements using one more combinations of ROCm meta packages:

- To use pre-built ROCm libraries and tools, include ROCm runtime packages in the installation step.
- To develop and build individual ROCm libraries and tools, include ROCm developer packages in the installation step.

4.2.2.3.2.3 ROCm runtime packages

Meta package	Description	Legacy use case[Page 47, 1]
rocm	All ROCm core packages, tools, and libraries.	rocm
rocm-hip-libraries	HIP libraries optimized for the AMD platform.	Legacy use case does not exist.
rocm-hip-runtime	Run HIP applications written for the AMD platform.	hip
rocm-language-runti	ROCm runtime environment for running applications on the AMD platform.	lrt
rocm-ml-libraries	Key machine learning libraries. Includes MIOpen.	mllib
rocm-opencl-runtim	Run OpenCL-based applications on the AMD platform.	opencl
Other package		
amdgpu-lib	For users of graphics applications which require the open source Mesa 3D graphics and multimedia libraries. This package is primarily used for Radeon GPUs.	graphics

4.2.2.3.2.4 ROCm developer packages

Meta package	Description	Legacy use case[1]
rocm-developer-tool	Debug and profile HIP applications.	rocmdevtools
rocm-hip-runtime-de rocm-hip-runtime-de	Develop applications on HIP or port from CUDA.	Legacy use case does not exist.
rocm-hip-sdk[3]	Develop or port HIP applications and libraries for the AMD platform.	hiplibsdk
rocm-ml-sdk	Develop and run machine learning applications for AMD.	mlsdk
rocm-opencl-sdk	Develop OpenCL-based applications for the AMD platform.	openclsdk
rocm-openmp-sdk	Develop OpenMP-based applications for the AMD software.	openmpsdk

4.2.2.3.3 Post-installation

Complete the Post-installation instructions.

4.2.2.3.4 Uninstalling

4.2.2.3.4.1 Uninstall ROCm meta packages

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

4.2.2.3.4.2 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
```

Important

To apply all settings, reboot your system.

4.2.2.4 Oracle Linux native installation

Caution

Ensure that the Installation prerequisites are met.

[1] Starting from ROCm 6.4.2, "Legacy use cases" in amdgpu-install are replaced by the equivalent meta package. In addition, the following amdgpu-install use cases: asan, rocmdev, multimedia, multimediasdk, amf, and workstation are deprecated.
[2] Use rocm-hip-runtime-dev for Debian/Ubuntu-based systems and rocm-hip-runtime-devel for RHEL/RPM-based systems.
[3] rocm-hip-sdk requires rocm-hip-runtime-devel.

> **Note**
>
> The following installation steps also apply when upgrading from a previous ROCm version.

4.2.2.4.1 Register ROCm repositories

OL 10.1

```
sudo tee /etc/yum.repos.d/rocm.repo <<EOF
[rocm]
name=ROCm 7.2.3 repository
baseurl=https://repo.radeon.com/rocm/el10/7.2.3/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key

[amdgraphics]
name=AMD Graphics 7.2.3 repository
baseurl=https://repo.radeon.com/graphics/7.2.3/el/10/main/x86_64/
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
sudo dnf clean all
```

OL 9.7

```
sudo tee /etc/yum.repos.d/rocm.repo <<EOF
[rocm]
name=ROCm 7.2.3 repository
baseurl=https://repo.radeon.com/rocm/el9/7.2.3/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key

[amdgraphics]
name=AMD Graphics 7.2.3 repository
baseurl=https://repo.radeon.com/graphics/7.2.3/el/9.7/main/x86_64/
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
sudo dnf clean all
```

OL 8.10

```
sudo tee /etc/yum.repos.d/rocm.repo <<EOF
[rocm]
name=ROCm 7.2.3 repository
baseurl=https://repo.radeon.com/rocm/el8/7.2.3/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key

[amdgraphics]
name=AMD Graphics 7.2.3 repository
baseurl=https://repo.radeon.com/graphics/7.2.3/el/8/main/x86_64/
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
sudo dnf clean all
```

4.2.2.4.2 Installing

4.2.2.4.2.1 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 AMD GPU Driver (amdgpu) support matrix.

4.2.2.4.2.2 Install ROCm

```
sudo dnf install rocm
```

ROCm installation can be tailored to your requirements using one more combinations of ROCm meta packages:

- To use pre-built ROCm libraries and tools, include ROCm runtime packages in the installation step.
- To develop and build individual ROCm libraries and tools, include ROCm developer packages in the installation step.

4.2.2.4.2.3 ROCm runtime packages

Meta package	Description	Legacy use case[1]
rocm	All ROCm core packages, tools, and libraries.	rocm
rocm-hip-libraries	HIP libraries optimized for the AMD platform.	Legacy use case does not exist.
rocm-hip-runtime	Run HIP applications written for the AMD platform.	hip
rocm-language-runti	ROCm runtime environment for running applications on the AMD platform.	lrt
rocm-ml-libraries	Key machine learning libraries. Includes MIOpen.	mllib
rocm-opencl-runtim	Run OpenCL-based applications on the AMD platform.	opencl
Other package		
amdgpu-lib	For users of graphics applications which require the open source Mesa 3D graphics and multimedia libraries. This package is primarily used for Radeon GPUs.	graphics

4.2.2.4.2.4 ROCm developer packages

Meta package	Description	Legacy use case[1]
rocm-developer-tool	Debug and profile HIP applications.	rocmdevtools
rocm-hip-runtime-d	Develop applications on HIP or port from CUDA.	Legacy use case does not exist.
rocm-hip-runtime-d		
rocm-hip-sdk[3]	Develop or port HIP applications and libraries for the AMD platform.	hiplibsdk
rocm-ml-sdk	Develop and run machine learning applications for AMD.	mlsdk
rocm-opencl-sdk	Develop OpenCL-based applications for the AMD platform.	openclsdk
rocm-openmp-sdk	Develop OpenMP-based applications for the AMD software.	openmpsdk

4.2.2.4.3 Post-installation

Complete the Post-installation instructions.

4.2.2.4.4 Uninstalling

4.2.2.4.4.1 Uninstall ROCm meta packages

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

[1] Starting from ROCm 6.4.2, "Legacy use cases" in amdgpu-install are replaced by the equivalent meta package. In addition, the following amdgpu-install use cases: asan, rocmdev, multimedia, multimediasdk, amf, and workstation are deprecated.

[2] Use rocm-hip-runtime-dev for Debian/Ubuntu-based systems and rocm-hip-runtime-devel for RHEL/RPM-based systems.

[3] rocm-hip-sdk requires rocm-hip-runtime-devel.

4.2.2.4.4.2 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
```

Important

To apply all settings, reboot your system.

4.2.2.5 Rocky Linux native installation

Caution

Ensure that the Installation prerequisites are met.

Note

The following installation steps also apply when upgrading from a previous ROCm version.

4.2.2.5.1 Registering ROCm repositories

Rocky 9.7

```
sudo tee /etc/yum.repos.d/rocm.repo <<EOF
[rocm]
name=ROCm 7.2.3 repository
baseurl=https://repo.radeon.com/rocm/el9/7.2.3/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key

[amdgraphics]
name=AMD Graphics 7.2.3 repository
baseurl=https://repo.radeon.com/graphics/7.2.3/el/9.7/main/x86_64/
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
sudo dnf clean all
```

4.2.2.5.2 Installing

4.2.2.5.2.1 Install kernel driver

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

For information about driver compatibility, see User and AMD GPU Driver (amdgpu) support matrix.

4.2.2.5.2.2 Install ROCm

```
sudo dnf install rocm
```

ROCm installation can be tailored to your requirements using one more combinations of ROCm meta packages:

- To use pre-built ROCm libraries and tools, include ROCm runtime packages in the installation step.
- To develop and build individual ROCm libraries and tools, include ROCm developer packages in the installation step.

4.2.2.5.2.3 ROCm runtime packages

Meta package	Description	Legacy use case[1]
rocm	All ROCm core packages, tools, and libraries.	rocm
rocm-hip-libraries	HIP libraries optimized for the AMD platform.	Legacy use case does not exist.
rocm-hip-runtime	Run HIP applications written for the AMD platform.	hip
rocm-language-runti	ROCm runtime environment for running applications on the AMD platform.	lrt
rocm-ml-libraries	Key machine learning libraries. Includes MIOpen.	mllib
rocm-opencl-runtim	Run OpenCL-based applications on the AMD platform.	opencl
Other package		
amdgpu-lib	For users of graphics applications which require the open source Mesa 3D graphics and multimedia libraries. This package is primarily used for Radeon GPUs.	graphics

[1] Starting from ROCm 6.4.2, "Legacy use cases" in amdgpu-install are replaced by the equivalent meta package. In addition, the following amdgpu-install use cases: asan, rocmdev, multimedia, multimediasdk, amf, and workstation are deprecated.

4.2.2.5.2.4 ROCm developer packages

Meta package	Description	Legacy use case[Page 52, 1]
rocm-developer-tool	Debug and profile HIP applications.	rocmdevtools
rocm-hip-runtime-de rocm-hip-runtime-de	Develop applications on HIP or port from CUDA.	Legacy use case does not exist.
rocm-hip-sdk[3]	Develop or port HIP applications and libraries for the AMD platform.	hiplibsdk
rocm-ml-sdk	Develop and run machine learning applications for AMD.	mlsdk
rocm-opencl-sdk	Develop OpenCL-based applications for the AMD platform.	openclsdk
rocm-openmp-sdk	Develop OpenMP-based applications for the AMD software.	openmpsdk

4.2.2.5.3 Post-installation

Complete the Post-installation instructions.

4.2.2.5.4 Uninstalling

4.2.2.5.4.1 Uninstall ROCm meta packages

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

4.2.2.5.4.2 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
```

Important

To apply all settings, reboot your system.

4.2.2.6 SUSE Linux Enterprise Server native installation

Caution

Ensure that the Installation prerequisites are met.

[2] Use rocm-hip-runtime-dev for Debian/Ubuntu-based systems and rocm-hip-runtime-devel for RHEL/RPM-based systems.
[3] rocm-hip-sdk requires rocm-hip-runtime-devel.

> **Note**
>
> The following installation steps also apply when upgrading from a previous ROCm version.

4.2.2.6.1 Registering ROCm repositories

SLES 15.7

```
sudo tee /etc/zypp/repos.d/rocm.repo <<EOF
[rocm]
name=ROCm 7.2.3 repository
baseurl=https://repo.radeon.com/rocm/zyp/7.2.3/main
enabled=1
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key

[amdgraphics]
name=AMD Graphics 7.2.3 repository
baseurl=https://repo.radeon.com/graphics/7.2.3/sle/15.7/main/x86_64/
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
sudo zypper refresh
```

4.2.2.6.2 Installing

4.2.2.6.2.1 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 AMD GPU Driver (amdgpu) support matrix.

4.2.2.6.2.2 Install ROCm

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

ROCm installation can be tailored to your requirements using one more combinations of ROCm meta packages:

- To use pre-built ROCm libraries and tools, include ROCm runtime packages in the installation step.
- To develop and build individual ROCm libraries and tools, include ROCm developer packages in the installation step.

4.2.2.6.2.3 ROCm runtime packages

Meta package	Description	Legacy use case[1]
rocm	All ROCm core packages, tools, and libraries.	rocm
rocm-hip-libraries	HIP libraries optimized for the AMD platform.	Legacy use case does not exist.
rocm-hip-runtime	Run HIP applications written for the AMD platform.	hip
rocm-language-runti	ROCm runtime environment for running applications on the AMD platform.	lrt
rocm-ml-libraries	Key machine learning libraries. Includes MIOpen.	mllib
rocm-opencl-runtim	Run OpenCL-based applications on the AMD platform.	opencl
Other package		
amdgpu-lib	For users of graphics applications which require the open source Mesa 3D graphics and multimedia libraries. This package is primarily used for Radeon GPUs.	graphics

4.2.2.6.2.4 ROCm developer packages

Meta package	Description	Legacy use case[1]
rocm-developer-tool	Debug and profile HIP applications.	rocmdevtools
rocm-hip-runtime-d	Develop applications on HIP or port from CUDA.	Legacy use case does not exist.
rocm-hip-runtime-d		
rocm-hip-sdk[3]	Develop or port HIP applications and libraries for the AMD platform.	hiplibsdk
rocm-ml-sdk	Develop and run machine learning applications for AMD.	mlsdk
rocm-opencl-sdk	Develop OpenCL-based applications for the AMD platform.	openclsdk
rocm-openmp-sdk	Develop OpenMP-based applications for the AMD software.	openmpsdk

4.2.2.6.3 Post-installation

Complete the Post-installation instructions.

4.2.2.6.4 Uninstalling

4.2.2.6.4.1 Uninstall ROCm meta packages

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

[1] Starting from ROCm 6.4.2, "Legacy use cases" in amdgpu-install are replaced by the equivalent meta package. In addition, the following amdgpu-install use cases: asan, rocmdev, multimedia, multimediasdk, amf, and workstation are deprecated.

[2] Use rocm-hip-runtime-dev for Debian/Ubuntu-based systems and rocm-hip-runtime-devel for RHEL/RPM-based systems.

[3] rocm-hip-sdk requires rocm-hip-runtime-devel.

4.2.2.6.4.2 Remove ROCm repositories

```
# Remove ROCm repositories
sudo zypper removerepo "rocm"
sudo zypper removerepo "amdgraphics"

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

Important

To apply all settings, reboot your system.

4.2.3 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.

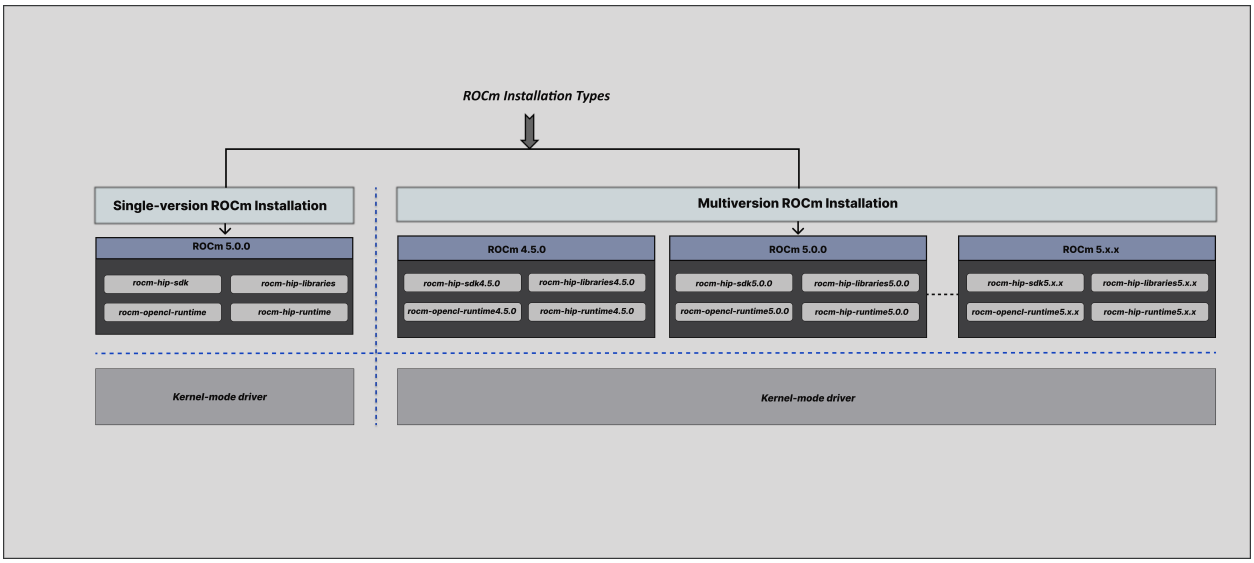
- Installing a single instance of the ROCm release on a system.
- Using non-versioned ROCm meta-packages.

See Quick start installation guide or Detailed install 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
- Rocky Linux
- SUSE Linux Enterprise Server

Uninstall

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

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

4.2.3.1 Ubuntu multi-version installation

Caution

Ensure that the Installation prerequisites are met.

4.2.3.1.1 Registering ROCm repositories

4.2.3.1.1.1 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.

4.2.3.1.1.2 Register packages

Ubuntu 24.04

```
# Note: There is NO trailing .0 in the patch version for repositories
for ver in 7.2.3 7.2; do
sudo tee --append /etc/apt/sources.list.d/rocm.list << EOF
deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/rocm/apt/$ver noble␣
↪main
EOF
done

sudo tee /etc/apt/preferences.d/rocm-pin-600 << EOF
Package: *
Pin: release o=repo.radeon.com
Pin-Priority: 600
EOF
sudo apt update
```

Ubuntu 22.04

```
# Note: There is NO trailing .0 in the patch version for repositories
for ver in 7.2.3 7.2; do
sudo tee --append /etc/apt/sources.list.d/rocm.list << EOF
deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/rocm/apt/$ver␣
↪jammy main
EOF
done

sudo tee /etc/apt/preferences.d/rocm-pin-600 << EOF
Package: *
```

(continues on next page)

(continued from previous page)

```
Pin: release o=repo.radeon.com
Pin-Priority: 600
EOF
sudo apt update
```

4.2.3.1.2 Installing

4.2.3.1.2.1 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 AMD GPU Driver (amdgpu) support matrix.

4.2.3.1.2.2 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 7.2.3 7.2.0; 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).

4.2.3.1.3 Post-installation

Complete the Post-installation instructions.

> **Tip**
>
> For a single-version installation of the latest ROCm version on Ubuntu, follow the steps in Ubuntu native installation in the ROCm documentation.

4.2.3.1.4 Uninstalling

4.2.3.1.4.1 Uninstall specific meta packages

```
# Note: There IS a trailing .0 in the patch version for packages
for ver in 7.2.3 7.2.0; do
    sudo apt autoremove rocm$ver
done
```

4.2.3.1.4.2 Uninstall ROCm packages

```
# Note: There IS a trailing .0 in the patch version for packages
for ver in 7.2.3 7.2.0; do
    sudo apt autoremove rocm-core$ver
done
```

4.2.3.1.4.3 Remove ROCm repositories

```
# Remove ROCm 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
```

Important

To apply all settings, reboot your system.

4.2.3.2 Debian multi-version installation

Caution

Ensure that the Installation prerequisites are met.

4.2.3.2.1 Registering ROCm repositories

4.2.3.2.1.1 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.

4.2.3.2.1.2 Register packages

Debian 13

```
# Note: There is NO trailing .0 in the patch version for repositories
for ver in 7.2.3 7.2; do
sudo tee --append /etc/apt/sources.list.d/rocm.list << EOF
deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/rocm/apt/$ver noble␣
↪main
EOF
done

sudo tee /etc/apt/preferences.d/rocm-pin-600 << EOF
Package: *
Pin: release o=repo.radeon.com
Pin-Priority: 600
EOF
sudo apt update
```

Debian 12

```
# Note: There is NO trailing .0 in the patch version for repositories
for ver in 7.2.3 7.2; do
sudo tee --append /etc/apt/sources.list.d/rocm.list << EOF
deb [arch=amd64 signed-by=/etc/apt/keyrings/rocm.gpg] https://repo.radeon.com/rocm/apt/$ver␣
↪jammy main
EOF
done

sudo tee /etc/apt/preferences.d/rocm-pin-600 << EOF
Package: *
Pin: release o=repo.radeon.com
Pin-Priority: 600
EOF
sudo apt update
```

4.2.3.2.2 Installing

4.2.3.2.2.1 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 AMD GPU Driver (amdgpu) support matrix.

4.2.3.2.2.2 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 7.2.3 7.2.0; 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).

4.2.3.2.3 Post-installation

Complete the Post-installation instructions.

> **Tip**
>
> For a single-version installation of the latest ROCm version on Debian, follow the steps in Debian native installation in the ROCm documentation.

4.2.3.2.4 Uninstalling

4.2.3.2.4.1 Uninstall specific meta packages

```
# Note: There IS a trailing .0 in the patch version for packages
for ver in 7.2.3 7.2.0; do
    sudo apt autoremove rocm$ver
done
```

4.2.3.2.4.2 Uninstall ROCm packages

```
# Note: There IS a trailing .0 in the patch version for packages
for ver in 7.2.3 7.2.0; do
    sudo apt autoremove rocm-core$ver
done
```

4.2.3.2.4.3 Remove ROCm repositories

```
# Remove ROCm 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
```

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

4.2.3.3 Red Hat Enterprise Linux multi-version installation

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

4.2.3.3.1 Registering ROCm repositories

RHEL 10.1

```
# Note: There is NO trailing .0 in the patch version for repositories
for ver in 7.2.3 7.2; do
sudo tee --append /etc/yum.repos.d/rocm.repo <<EOF
[rocm-$ver]
name=ROCm $ver repository
baseurl=https://repo.radeon.com/rocm/el10/$ver/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
done
sudo dnf clean all
```

RHEL 10.0

```
# Note: There is NO trailing .0 in the patch version for repositories
for ver in 7.2.3 7.2; do
sudo tee --append /etc/yum.repos.d/rocm.repo <<EOF
[rocm-$ver]
name=ROCm $ver repository
baseurl=https://repo.radeon.com/rocm/el10/$ver/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
done
sudo dnf clean all
```

RHEL 9.7

```
# Note: There is NO trailing .0 in the patch version for repositories
for ver in 7.2.3 7.2; do
sudo tee --append /etc/yum.repos.d/rocm.repo <<EOF
[rocm-$ver]
name=ROCm $ver repository
baseurl=https://repo.radeon.com/rocm/el9/$ver/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
done
sudo dnf clean all
```

RHEL 9.6

```
# Note: There is NO trailing .0 in the patch version for repositories
for ver in 7.2.3 7.2; do
sudo tee --append /etc/yum.repos.d/rocm.repo <<EOF
[rocm-$ver]
name=ROCm $ver repository
baseurl=https://repo.radeon.com/rocm/el9/$ver/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
done
sudo dnf clean all
```

RHEL 9.4

```
# Note: There is NO trailing .0 in the patch version for repositories
for ver in 7.2.3 7.2; do
sudo tee --append /etc/yum.repos.d/rocm.repo <<EOF
[rocm-$ver]
name=ROCm $ver repository
baseurl=https://repo.radeon.com/rocm/el9/$ver/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
done
sudo dnf clean all
```

RHEL 8.10

```
# Note: There is NO trailing .0 in the patch version for repositories
for ver in 7.2.3 7.2; do
sudo tee --append /etc/yum.repos.d/rocm.repo <<EOF
[rocm-$ver]
name=ROCm $ver repository
baseurl=https://repo.radeon.com/rocm/el8/$ver/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
done
sudo dnf clean all
```

4.2.3.3.2 Installing

4.2.3.3.2.1 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 AMD GPU Driver (amdgpu) support matrix.

4.2.3.3.2.2 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 7.2.3 7.2.0; 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).

4.2.3.3.3 Post-installation

Complete the Post-installation instructions.

> **Tip**
>
> For a single-version installation of the latest ROCm version on RHEL, follow the steps in Red Hat Enterprise Linux native installation in the ROCm documentation.

4.2.3.3.4 Uninstalling

4.2.3.3.4.1 Uninstall specific meta packages

```
# Note: There IS a trailing .0 in the patch version for packages
for ver in 7.2.3 7.2.0; do
   sudo dnf remove rocm$ver
done
```

4.2.3.3.4.2 Uninstall ROCm packages

```
# Note: There IS a trailing .0 in the patch version for packages
for ver in 7.2.3 7.2.0; do
   sudo dnf remove rocm-core$ver amdgpu-core$ver
done
```

4.2.3.3.4.3 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
```

Important

To apply all settings, reboot your system.

4.2.3.4 Oracle Linux multi-version installation

Caution

Ensure that the Installation prerequisites are met.

4.2.3.4.1 Register ROCm repositories

OL 10.1

```
# Note: There is NO trailing .0 in the patch version for repositories
for ver in 7.2.3 7.2; do
sudo tee --append /etc/yum.repos.d/rocm.repo <<EOF
[rocm-$ver]
name=ROCm $ver repository
baseurl=https://repo.radeon.com/rocm/el10/$ver/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
done
sudo dnf clean all
```

OL 9.7

```
# Note: There is NO trailing .0 in the patch version for repositories
for ver in 7.2.3 7.2; do
sudo tee --append /etc/yum.repos.d/rocm.repo <<EOF
[rocm-$ver]
name=ROCm $ver repository
baseurl=https://repo.radeon.com/rocm/el9/$ver/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
```

(continues on next page)

(continued from previous page)

```
done
sudo dnf clean all
```

OL 8.10

```
# Note: There is NO trailing .0 in the patch version for repositories
for ver in 7.2.3 7.2; do
sudo tee --append /etc/yum.repos.d/rocm.repo <<EOF
[rocm-$ver]
name=ROCm $ver repository
baseurl=https://repo.radeon.com/rocm/el8/$ver/main
enabled=1
priority=50
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
done
sudo dnf clean all
```

4.2.3.4.2 Installing

4.2.3.4.2.1 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 AMD GPU Driver (amdgpu) support matrix.

4.2.3.4.2.2 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.

OL 10.1

```
# Note: There IS a trailing .0 in the patch version for packages
for ver in 7.2.3 7.2.0; do
    sudo dnf install rocm$ver
done
```

OL 9.7

```
# Note: There IS a trailing .0 in the patch version for packages
for ver in 7.2.3 7.2.0; do
    sudo dnf install rocm$ver
done
```

OL 8.10

```
# Note: There IS a trailing .0 in the patch version for packages
for ver in 7.2.3 7.2.0; 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).

4.2.3.4.3 Post-installation

Complete the Post-installation instructions.

> **Tip**
>
> For a single-version installation of the latest ROCm version on OL, follow the steps in Oracle Linux native installation in the ROCm documentation.

4.2.3.4.4 Uninstalling

4.2.3.4.4.1 Uninstall specific meta packages

OL 10.1

```
# Note: There IS a trailing .0 in the patch version for packages
for ver in 7.2.3 7.2.0; do
    sudo dnf remove rocm$ver
done
```

OL 9.7

```
# Note: There IS a trailing .0 in the patch version for packages
for ver in 7.2.3 7.2.0; do
    sudo dnf remove rocm$ver
done
```

OL 8.10

```
# Note: There IS a trailing .0 in the patch version for packages
for ver in 7.2.3 7.2.0; do
    sudo dnf remove rocm$ver
done
```

4.2.3.4.4.2 Uninstall ROCm packages

OL 10.1

```
# Note: There IS a trailing .0 in the patch version for packages
for ver in 7.2.3 7.2.0; do
      sudo dnf remove rocm-core$ver amdgpu-core$ver
done
```

OL 9.7

```
# Note: There IS a trailing .0 in the patch version for packages
for ver in 7.2.3 7.2.0; do
      sudo dnf remove rocm-core$ver amdgpu-core$ver
done
```

OL 8.10

```
# Note: There IS a trailing .0 in the patch version for packages
for ver in 7.2.3 7.2.0; do
      sudo dnf remove rocm-core$ver amdgpu-core$ver
done
```

4.2.3.4.4.3 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
```

Important

To apply all settings, reboot your system.

4.2.3.5 Rocky Linux multi-version installation

Caution

Ensure that the Installation prerequisites are met.

4.2.3.5.1 Registering ROCm repositories

Rocky 9.7

```
# Note: There is NO trailing .0 in the patch version for repositories
for ver in 7.2.3 7.2; do
sudo tee --append /etc/yum.repos.d/rocm.repo <<EOF
[rocm-$ver]
name=ROCm $ver repository
baseurl=https://repo.radeon.com/rocm/el9/$ver/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
done
sudo dnf clean all
```

4.2.3.5.2 Installing

4.2.3.5.2.1 Install kernel driver

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

For information about driver compatibility, see User and AMD GPU Driver (amdgpu) support matrix.

4.2.3.5.2.2 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 7.2.3 7.2.0; 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).

4.2.3.5.3 Post-installation

Complete the Post-installation instructions.

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

4.2.3.5.4 Uninstalling

4.2.3.5.4.1 Uninstall specific meta packages

```
# Note: There IS a trailing .0 in the patch version for packages
for ver in 7.2.3 7.2.0; do
   sudo dnf remove rocm$ver
done
```

4.2.3.5.4.2 Uninstall ROCm packages

```
# Note: There IS a trailing .0 in the patch version for packages
for ver in 7.2.3 7.2.0; do
   sudo dnf remove rocm-core$ver amdgpu-core$ver
done
```

4.2.3.5.4.3 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
```

Important

To apply all settings, reboot your system.

4.2.3.6 SUSE Linux Enterprise Server multi-version installation

Caution

Ensure that the Installation prerequisites are met.

4.2.3.6.1 Registering ROCm repositories

SLES 15.7

```
# Note: There is NO trailing .0 in the patch version for repositories
for ver in 7.2.3 7.2; do
sudo tee --append /etc/zypp/repos.d/rocm.repo <<EOF
[rocm-$ver]
name=ROCm $ver repository
baseurl=https://repo.radeon.com/rocm/zyp/$ver/main
enabled=1
gpgcheck=1
gpgkey=https://repo.radeon.com/rocm/rocm.gpg.key
EOF
done
sudo zypper refresh
```

4.2.3.6.2 Installing

4.2.3.6.2.1 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 AMD GPU Driver (amdgpu) support matrix.

4.2.3.6.2.2 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 7.2.3 7.2.0; 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).

4.2.3.6.3 Post-installation

Complete the Post-installation instructions.

> **Tip**
>
> For a single-version installation of the latest ROCm version on SLES, follow the steps in SUSE Linux Enterprise Server native installation in the ROCm documentation.

4.2.3.6.4 Uninstalling

4.2.3.6.4.1 Uninstall specific meta packages

```
# Note: There IS a trailing .0 in the patch version for packages
for ver in 7.2.3 7.2.0; do
   sudo zypper remove rocm$ver
done
```

4.2.3.6.4.2 Uninstall ROCm packages

```
# Note: There IS a trailing .0 in the patch version for packages
for ver in 7.2.3 7.2.0; do
   sudo zypper remove rocm-core$ver amdgpu-core$ver
done
```

4.2.3.6.4.3 Remove ROCm repositories

```
# Remove ROCm repositories
# Note: There is NO trailing .0 in the patch version for repositories
for ver in 7.2.3 7.2; do
   sudo zypper removerepo "rocm-$ver"
done

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

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

4.2.4 ROCm Runfile Installer

The ROCm Runfile Installer installs ROCm, the AMD GPU 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 AMD GPU 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 AMD GPU Driver installation (requires the prior installation of dependencies)
- Packageless ROCm and AMD GPU Driver install without native package management
- A single self-contained installer for all ROCm and AMD GPU Driver software
- Configurable installation location for the ROCm install
- Basic “tarball-like” extraction of ROCm content to a target location.

4.2.4.1 Prerequisites

The ROCm Runfile Installer requires the following configuration:

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

4.2.4.2 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 AMD GPU 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

4.2.4.2.1 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 AMD GPU Driver. This list, which indicates all packages required for the ROCm runtime or the AMD GPU Driver, is displayed and saved to a file named deps_list.txt in the root rocm-installer directory. The required libraries, frameworks, and other components within the required packages must be on the system when running ROCm or installing the AMD GPU Driver. Users can manually install the required packages in the list using any method.

The list of packages that is displayed and saved is not directly usable by the package manager. Each line indicates the name of the package or versioning required to meet the dependency requirement. Some dependencies might show multiple packages separated by the | symbol, meaning that any package from the list satisfies the package dependency. For example, the libgcc-dev dependency might be libgcc-5-dev | libgcc-7-dev, meaning that either libgcc-5-dev or libgcc-7-dev can be installed to satisfy the corresponding dependency.

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.

4.2.4.2.2 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 AMD GPU 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.

4.2.4.3 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, 9.6, 9.7, 10.0, 10.1
- SLES: 15.7
- Debian: 12, 13
- Oracle Linux: 8.10, 9.7, 10.0
- Rocky Linux: 9.7

Oracle Linux 8 and 9 use the corresponding RHEL 8 and 9 builds. Debian 12 and 13 use the Ubuntu 22.04 and 24.04 builds. Rocky Linux 9 uses the RHEL 9 builds. The following table maps the supported Linux distributions to the Runfile Installer builds they use:

Linux distribution	Runfile build
Ubuntu	Ubuntu
RHEL	RHEL
SLES	SLES
Debian 12	Ubuntu 22.04
Debian 13	Ubuntu 24.04
Oracle Linux	RHEL
Rocky Linux 9	RHEL 9

4.2.4.4 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.

4.2.4.4.1 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 7.2.3 of the ROCm Runfile Installer for Ubuntu release 22.04:

```
wget https://repo.radeon.com/rocm/installer/rocm-runfile-installer/rocm-rel-7.2.3/ubuntu/22.04/rocm-
↪installer_1.2.8.70203-61-90~22.04.run
```

4.2.4.4.2 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.

4.2.4.5 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.

4.2.4.6 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
```

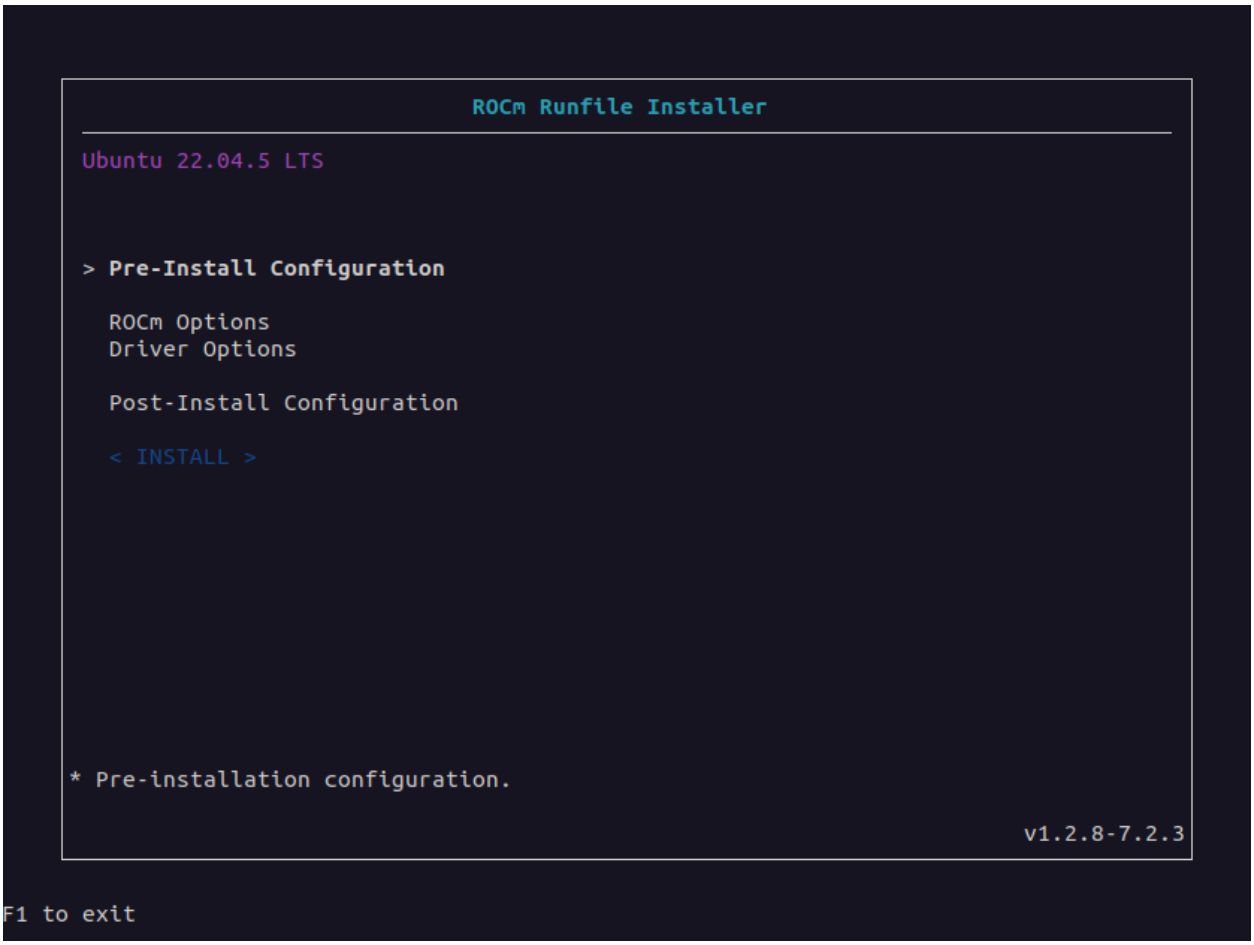
4.2.4.6.1 GUI

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.

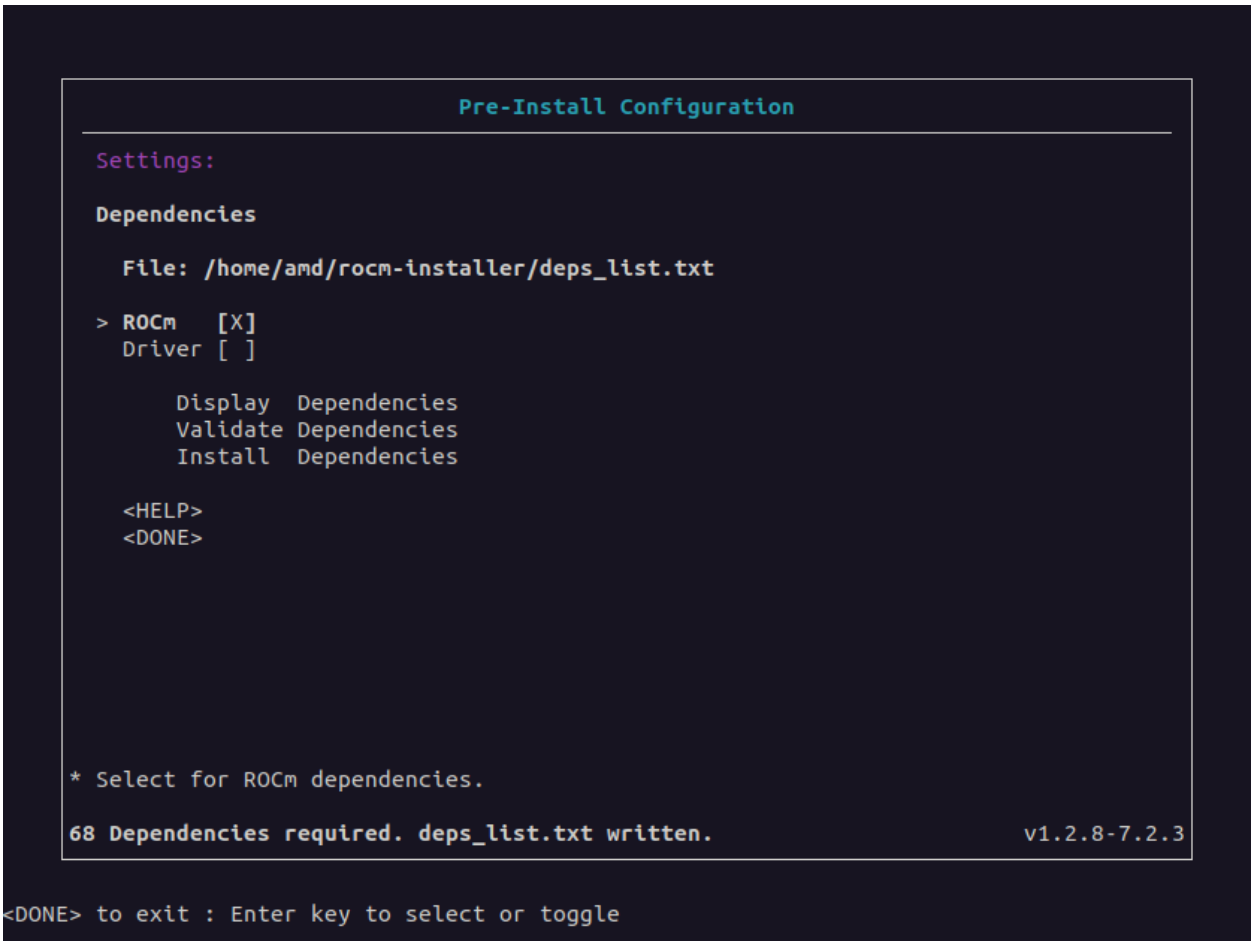
4.2.4.6.1.1 Main menu

The Main menu is the installation starting point.



4.2.4.6.1.2 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 AMD GPU 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 AMD GPU 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 AMD GPU 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.

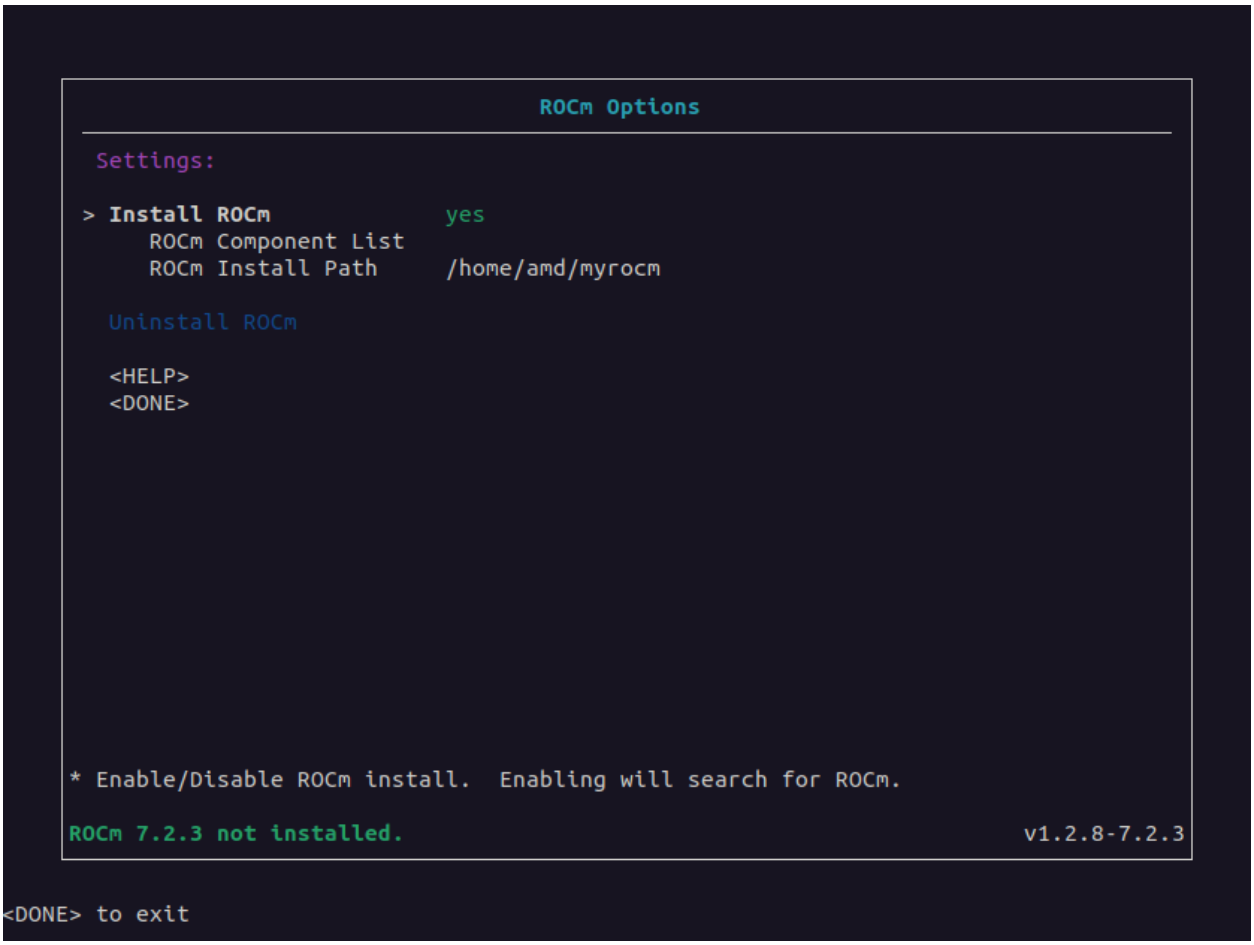
- Install Dependencies

  If the installer is running on the system where ROCm or the AMD GPU 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.

4.2.4.6.1.3 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.

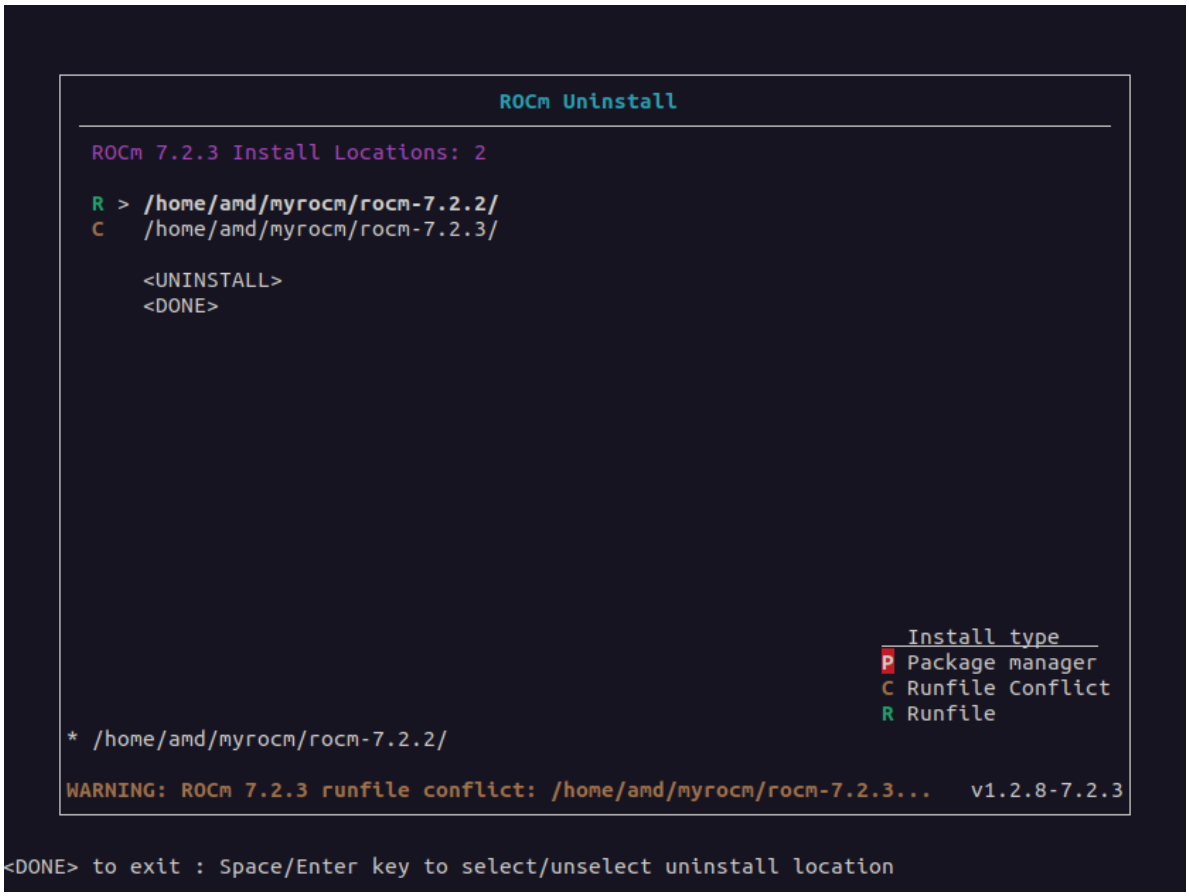
  When ROCm Install Path is set to a new path, the installer validates the new directory location. If the directory exists, the ROCm installation can proceed. If an invalid location is specified, the ROCm installation will not be allowed.

- 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:

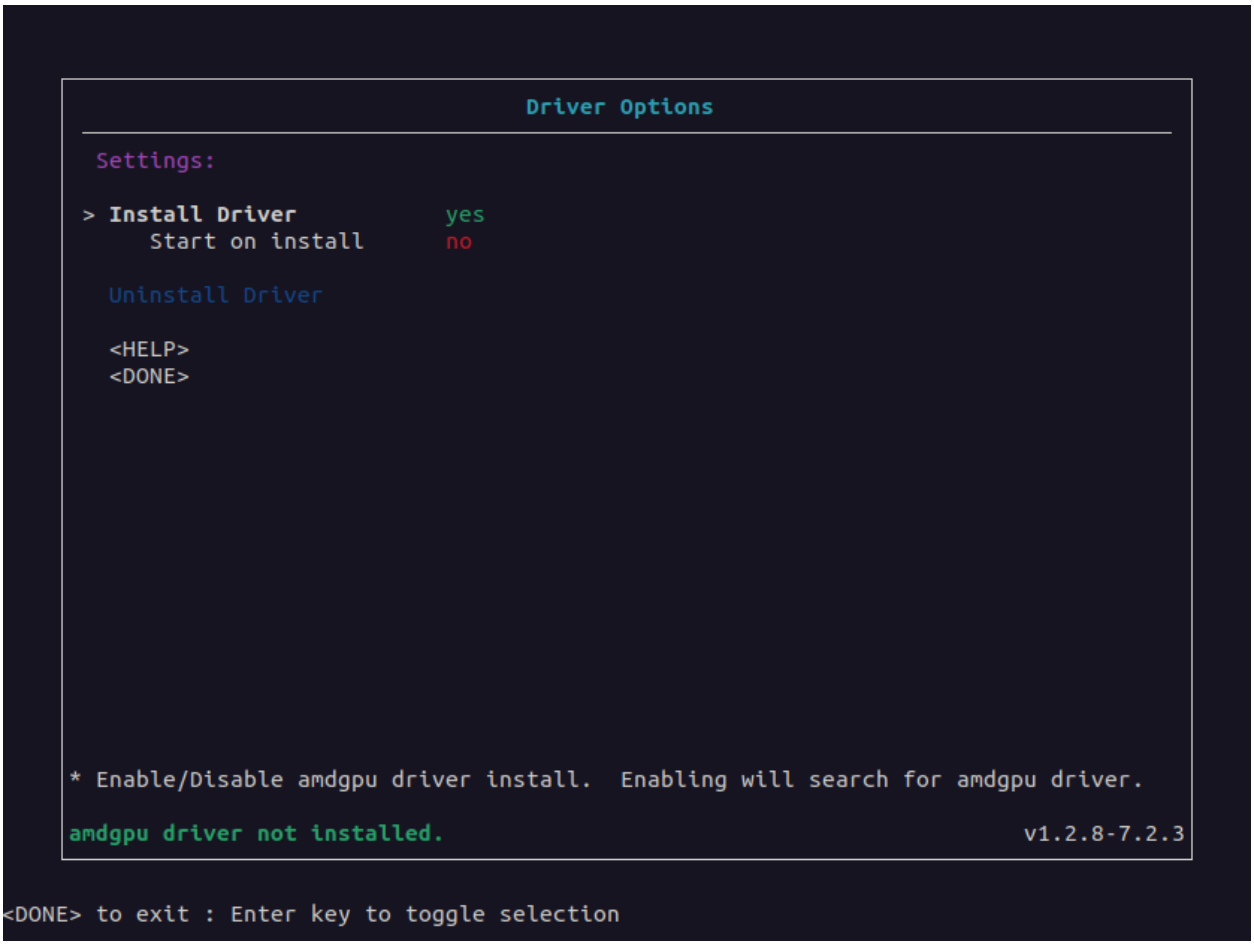
- P (Package manager): Package manager installation that matches the ROCm version for the Runfile Installer. In this case, uninstall is not allowed.
- C (Runfile conflict): Runfile installation that matches the ROCm version for the Runfile Installer. The conflicting installation can be uninstalled.
- 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.



4.2.4.6.1.4 Driver Options menu

The Driver Options menu can include or exclude the AMD GPU Driver from the installation.



- Install Driver

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

  > **Note**
  >
  > The AMD GPU Driver is installed for the currently running Linux kernel version. To install the AMD GPU Driver for a different kernel, reboot to the specific Linux kernel and reinstall the AMD GPU Driver using the runfile.

- Start on install

  If the Start on install option is enabled, the Runfile installer uses modprobe to automatically launch the AMD GPU Driver after installation. If a pre-existing AMD GPU 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.

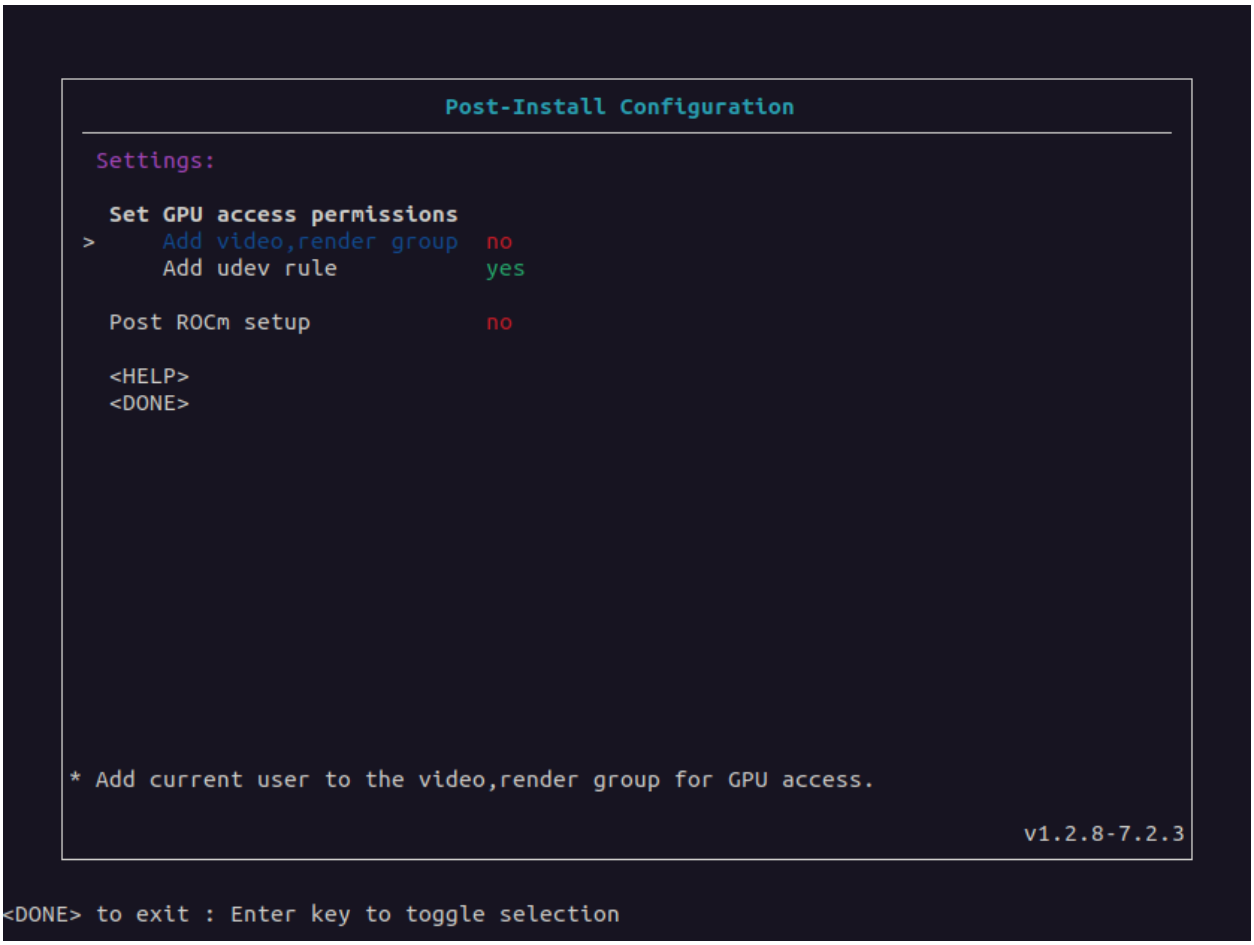
- Uninstall Driver

  This field is only available if a previous Runfile install of the AMD GPU Driver is discovered on the install system. If a Runfile installation of the AMD GPU Driver is detected, select Uninstall Driver

to remove it. Package manager installs of the AMD GPU Driver cannot be uninstalled by the Runfile installer and must be uninstalled manually using the package management application.

4.2.4.6.1.5 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.

4.2.4.6.2 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 AMD GPU 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 AMD GPU Driver options:
   a. Enter the Driver Options menu.
   b. Set Install Driver to yes to include the AMD GPU 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.

4.2.4.7 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.
- Runfile extraction options
  - untar <directory>: Extract only the ROCm installation components from the .run file tarball to <directory>.
  - untar <directory> verbose: Extract only the ROCm installation components from the .run file tarball to <directory> and verbosely list the files that are processed.
- 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 AMD GPU Driver install.
  - force: Force the ROCm and AMD GPU 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 AMD GPU 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 AMD GPU 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 AMD GPU 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 AMD GPU Driver installations:

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

4.2.4.7.1 Command line interface

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

4.2.4.7.1.1 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 AMD GPU 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.

4.2.4.7.1.2 Runfile extraction options

The ROCm Runfile Installer extraction process can be run manually to generate a “tarball-like” extraction of only the ROCm component contents. The untar operation mode bypasses all checksum and installer execution and only outputs the ROCm content to a user-specified directory. Specifically, this mode of

operation allows users without system administrator (sudo) access to easily set up ROCm in local “home” directories. System administrator access is typically required to use the Runfile Installer otherwise.

Basic extraction is performed using the untar command line option:

- untar <directory>
- untar <directory> verbose

  The untar option is a command-line argument that extracts the ROCm component contents to the specified <directory> location. The output <directory> must be an absolute or relative path to a pre-existing directory on the system. The optional verbose argument can be added to the untar option to output the list of files being extracted.

  For example, to untar the ROCm content to a user directory called amd/myrocm, the command line is as follows:

```
bash rocm-installer.run untar /home/amd/myrocm
```

  For verbose extraction, the command line is as follows:

```
bash rocm-installer.run untar /home/amd/myrocm verbose
```

  After the untar extraction process completes, the content is written to a new rocm-version directory within the specified <directory>. The rocm-version directory is named according to the version of ROCm included in the Runfile Installer. For example, if the ROCm 7.2.3 Runfile Installer is extracted using the untar command, the output directory is named rocm-7.2.3.

  The extracted ROCm directory only contains the content for ROCm components and is not configured using the standard ROCm post-installation procedure. Advanced users might want to configure the ROCm content manually. However, as an option for some users, the untar command auto-generates a script that can be used for basic ROCm post-install configuration as part of the untarring process. The setup-modules script is generated and located in the root rocm-version directory.

  For example, after untarring ROCm 7.2.3 to the amd/myrocm user directory, the script can be found in the following location:

```
/home/amd/myrocm/rocm-7.2.3/setup-modules-7.2.3.sh
```

  > **Note**
  >
  > The setup-modules script requires system administrator access to execute and, therefore, might not be suitable for all users using the untar command line option.

  The setup-modules script sets up a ROCm module in the specified <directory> using environment-modules, which are installed when the script is run. After the setup-modules script has set up the environment-modules and additional symbolic links, the associated ROCm module can be loaded, allowing it to be used as part of ROCm. To set up and load the ROCm module for an untar extraction, where <version> is the Runfile Installer ROCm version, use the following steps:

```
cd <directory>/rocm-<version>
./setup-modules-<version>.sh
source /etc/profile.d/modules.sh
module load rocm/<version>
```

  For example, for a ROCm 7.2.3 Runfile Installer untar extraction to the amd/myrocm user directory, use the folllowing commands to set up and load a ROCm module:

```
cd /home/amd/myrocm/rocm-7.2.3
./setup-modules-7.2.3.sh
source /etc/profile.d/modules.sh
module load rocm/7.2.3
```

After the ROCm module is loaded, ROCm is functionally ready for use, provided all ROCm dependencies have been installed on the system. See the Dependency requirements section for more details on how to install ROCm dependencies that are not already installed.

4.2.4.7.1.3 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 AMD GPU 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 AMD GPU 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 AMD GPU 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 AMD GPU 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 AMD GPU Driver, the command line is as follows:

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

  To install the dependencies and both ROCm and the AMD GPU 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, AMD GPU Driver 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
  > ```

4.2.4.7.1.4 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. The AMD GPU Driver is installed for the currently running Linux kernel version. To install the AMD GPU Driver for a different kernel, reboot to the specific Linux kernel and reinstall the AMD GPU Driver using the runfile.

  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 AMD GPU 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 AMD GPU Driver installation.

  > **Note**
  >
  > This option must be in the list of .run <options> for AMD GPU Driver installation.

  For example, to install the AMD GPU 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.

4.2.4.7.1.5 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 7.2.3, then rocm-install-path must indicate the path to a ROCm 7.2.3 Runfile installation.

  To use the postrocm argument separately from the initial install of ROCm 7.2.3 to /home/amd/myrocm, run:

  ```
  bash rocm-installer.run target="/home/amd/myrocm/rocm-7.2.3" 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.

4.2.4.7.1.6 Uninstall options

These options configure the ROCm Runfile Installer to uninstall a previous ROCm or AMD GPU 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 AMD GPU Driver installation.

To uninstall the AMD GPU Driver from the system, use the following command line:

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

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

4.2.4.7.1.7 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
  ```

4.2.4.8 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.

4.3 Post-installation instructions

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

4.3.1 Environment Configuration

4.3.1.1 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
```

4.3.1.2 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/7.2.3
```

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

```
module switch rocm/7.2.3
```

> **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-7.2.3/bin
```

4.3.1.3 3. Configure LD_LIBRARY_PATH

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

```
export LD_LIBRARY_PATH=/opt/rocm-7.2.3/lib
```

4.3.2 Install verification

Once ROCm has been configured, validate the installation.

4.3.2.1 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
```

Rocky

```
dnf list installed
```

SLES

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

4.3.2.2 2. Verify the ROCm installation

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

rocminfo

```
rocminfo | grep -i "Marketing Name:"
```

Example output:

```
Marketing Name:          AMD EPYC 9654 96-Core Processor
Marketing Name:          AMD EPYC 9654 96-Core Processor
Marketing Name:          AMD Instinct MI300X
```

clinfo

```
clinfo | grep -i "Board name:"
```

Example output:

```
Board name:                              AMD Instinct MI300X
```

amd-smi

```
amd-smi version
```

Example output:

```
AMDSMI Tool: 26.2.2+c2d9476115 | AMDSMI Library version: 26.2.2 | ROCm version: 7.2.3 | amdgpu␣
↪version: 6.16.13 | hsmp version: N/A
```

4.3.3 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-7.2.3/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

FIVE

PYTORCH ON ROCM INSTALLATION

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

This topic covers setup instructions and the necessary files to build, test, and run PyTorch with ROCm support in a Docker environment. To learn more about PyTorch on ROCm, including its use cases, recommendations, as well as hardware and software compatibility, see PyTorch compatibility.

5.1 Install PyTorch

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

- Use a prebuilt Docker image with PyTorch pre-installed (recommended)
  - Docker image support
- Use a wheels package
- Use the PyTorch upstream Dockerfile

5.1.1 Use a prebuilt Docker image with PyTorch pre-installed

The recommended setup to get a PyTorch environment is through Docker, as it avoids potential installation issues. The tested, prebuilt image includes PyTorch, ROCm, and other dependencies. See Docker image support. To install ROCm on bare metal, follow ROCm installation overview.

1. Download the latest public PyTorch Docker image.

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

   Important

   The rocm/pytorch:latest tag points to a Docker image with the latest ROCm-tested release of PyTorch.

   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 \
   ```

(continues on next page)

(continued from previous page)

```
  --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.

5.1.2 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 7.2.3.

PyTorch 2.9.1

Python 3.12

Docker pull tag

```
docker pull rocm/pytorch:rocm7.2.3_ubuntu24.04_py3.12_pytorch_release_2.9.1
```

Additional software components

Ubuntu	Apex	torchvision	UCX	Open MPI
24.04	1.9.0+rocm7.2.3	0.24.0	1.16.0+ds-5ubuntu1	4.1.6-7ubuntu2

See rocm/pytorch:rocm7.2.3_ubuntu24.04_py3.12_pytorch_release_2.9.1 on Docker Hub.

Python 3.10

Docker pull tag

```
docker pull rocm/pytorch:rocm7.2.3_ubuntu22.04_py3.10_pytorch_release_2.9.1
```

Additional software components

Ubuntu	Apex	torchvision	UCX	Open MPI
22.04	1.9.0+rocm7.2.3	0.24.0	1.12.1~rc2-1	4.1.2-2ubuntu1

See rocm/pytorch:rocm7.2.3_ubuntu22.04_py3.10_pytorch_release_2.9.1 on Docker Hub.

PyTorch 2.8.0

Python 3.12

Docker pull tag

```
docker pull rocm/pytorch:rocm7.2.3_ubuntu24.04_py3.12_pytorch_release_2.8.0
```

Additional software components

Ubuntu	Apex	torchvision	UCX	Open MPI
24.04	1.8.0+rocm7.2.3	0.23.0	1.16.0+ds-5ubuntu1	4.1.6-7ubuntu2

See rocm/pytorch:rocm7.2.3_ubuntu24.04_py3.12_pytorch_release_2.8.0 on Docker Hub.

Python 3.10

Docker pull tag

```
docker pull rocm/pytorch:rocm7.2.3_ubuntu22.04_py3.10_pytorch_release_2.8.0
```

Additional software components

Ubuntu	Apex	torchvision	UCX	Open MPI
22.04	1.8.0+rocm7.2.3	0.23.0	1.12.1~rc2-1	4.1.2-2ubuntu1

See rocm/pytorch:rocm7.2.3_ubuntu22.04_py3.10_pytorch_release_2.8.0 on Docker Hub.

PyTorch 2.7.1

Python 3.12

Docker pull tag

```
docker pull rocm/pytorch:rocm7.2.3_ubuntu24.04_py3.12_pytorch_release_2.7.1
```

Additional software components

Ubuntu	Apex	torchvision	UCX	Open MPI
24.04	1.7.0+rocm7.2.3	0.22.1	1.16.0+ds-5ubuntu1	4.1.6-7ubuntu2

See rocm/pytorch:rocm7.2.3_ubuntu24.04_py3.12_pytorch_release_2.7.1 on Docker Hub.

Python 3.10

Docker pull tag

```
docker pull rocm/pytorch:rocm7.2.3_ubuntu22.04_py3.10_pytorch_release_2.7.1
```

Additional software components

Ubuntu	Apex	torchvision	UCX	Open MPI
22.04	1.7.0+rocm7.2.3	0.22.1	1.12.1~rc2-1	4.1.2-2ubuntu1

See rocm/pytorch:rocm7.2.3_ubuntu22.04_py3.10_pytorch_release_2.7.1 on Docker Hub.

5.1.3 Use 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.

Setting up the environment for the wheel installation

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 instructions 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
```

- Install torch, torchvision, and torchaudio, as specified in the installation matrix.

```
pip3 install --pre torch torchvision torchaudio --index-url https://download.pytorch.org/whl/nightly/
↪rocm7.2
```

> **Note**
>
> The above command uses the ROCm 7.2 PyTorch wheel. If you want a different version of ROCm, modify the command accordingly.

5.1.4 Build PyTorch from source

Use the rocm/pytorch:latest image, uninstall the preinstalled PyTorch package, and rebuild PyTorch from source. This ensures compatibility with your specific ROCm version, GPU architecture, and project requirements.

1. Download the latest PyTorch Docker image.

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

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
```

3. Uninstall the pre-installed PyTorch inside the container. Otherwise, the prebuilt ROCm PyTorch from the container might conflict with your source build.

```
pip3 uninstall -y torch torchvision torchaudio
```

4. Clone the PyTorch repository.

```
cd ~
git clone https://github.com/pytorch/pytorch.git
cd pytorch
git submodule update --init --recursive
```

5. (Optional) Set your ROCm architecture.

   By default, PyTorch builds for a broad set of AMD architectures. To speed up compilation, you can target only your GPU architecture.

   To determine your architecture:

   ```
   rocminfo | grep gfx
   ```

   Then set the PYTORCH_ROCM_ARCH environment variable:

   ```
   export PYTORCH_ROCM_ARCH=<uarch>
   ```

   Replace <uarch> with the result from rocminfo (for example, gfx90a, gfx1030). See System requirements (Linux) for the list of AMD GPU architectures.

6. Build and install PyTorch following the instructions in https://github.com/pytorch/pytorch?tab=readme-ov-file#install-pytorch.

5.1.5 Use 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
```

5.2 Test 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.

5.3 Run a 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.

5.3.1 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%)
```

5.3.2 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
```

5.4 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__)' ) )
llvm-readobj --offloading $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 Use 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

SIX

TENSORFLOW ON ROCM INSTALLATION

TensorFlow is an open-source library for solving machine learning, deep learning, and AI problems.

This topic covers setup instructions and the necessary files to build, test, and run TensorFlow with ROCm support in a Docker environment. To learn more about TensorFlow on ROCm, including its use cases, recommendations, as well as hardware and software compatibility, see TensorFlow compatibility.

> **Note**
>
> As of ROCm 6.1.0, tensorflow-rocm packages are found at https://repo.radeon.com/rocm/manylinux. Prior to ROCm 6.1.0, packages were found at https://pypi.org/project/tensorflow-rocm.

ROCm version	TensorFlow version
7.2.x	2.20.0, 2.19.1, 2.18.1
7.1.x	2.20.0, 2.19.1, 2.18.1
7.0.x	2.19.1, 2.18.1, 2.17.1
6.4.x	2.18.1, 2.17.1, 2.16.2
6.3.x	2.17.0, 2.16.2 2.15.1

6.1 Install TensorFlow

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

- Use a prebuilt Docker image with TensorFlow pre-installed (recommended)
  - Docker image support
- Use a wheels package

6.1.1 Use a prebuilt Docker image with TensorFlow pre-installed

The recommended setup to get a TensorFlow environment is through Docker, as it avoids potential installation issues. The tested, prebuilt image includes TensorFlow, ROCm, and other dependencies. See Docker image support. To install ROCm on bare metal, follow ROCm installation overview.

1. Download 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 \
    /bin/bash
```

6.1.2 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 7.2.3.

tensorflow-rocm 2.20.0

Python 3.12

Docker pull tag

```
docker pull rocm/tensorflow:rocm7.2.3-py3.12-tf2.20-dev
```

Additional software components

Ubuntu	Tensorboard
24.04	2.20.0

See rocm/tensorflow:rocm7.2.3-py3.12-tf2.20-dev on Docker Hub.

Python 3.10

Docker pull tag

```
docker pull rocm/tensorflow:rocm7.2.3-py3.10-tf2.20-dev
```

Additional software components

Ubuntu	Tensorboard
22.04	2.20.0

See rocm/tensorflow:rocm7.2.3-py3.10-tf2.20-dev on Docker Hub.

tensorflow-rocm 2.19.1

Python 3.12

Docker pull tag

```
docker pull rocm/tensorflow:rocm7.2.3-py3.12-tf2.19-dev
```

Additional software components

Ubuntu	Tensorboard
24.04	2.19.0

See rocm/tensorflow:rocm7.2.3-py3.12-tf2.19-dev on Docker Hub.

Python 3.10

Docker pull tag

```
docker pull rocm/tensorflow:rocm7.2.3-py3.10-tf2.19-dev
```

Additional software components

Ubuntu	Tensorboard
22.04	2.19.0

See rocm/tensorflow:rocm7.2.3-py3.10-tf2.19-dev on Docker Hub.

tensorflow-rocm 2.18.1

Python 3.12

Docker pull tag

```
docker pull rocm/tensorflow:rocm7.2.3-py3.12-tf2.18-dev
```

Additional software components

Ubuntu	Tensorboard
24.04	2.18.0

See rocm/tensorflow:rocm7.2.3-py3.12-tf2.18-dev on Docker Hub.

Python 3.10

Docker pull tag

```
docker pull rocm/tensorflow:rocm7.2.3-py3.10-tf2.18-dev
```

Additional software components

Ubuntu	Tensorboard
22.04	2.18.0

See rocm/tensorflow:rocm7.2.3-py3.10-tf2.18-dev on Docker Hub.

6.1.3 Use 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.

6.2 Test 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'
```

6.3 Run a 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'])
```

(continues on next page)

(continued from previous page)

```
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

SEVEN

JAX ON ROCM INSTALLATION

JAX is a library for array-oriented numerical computation (similar to NumPy), with automatic differentiation and just-in-time (JIT) compilation to enable high-performance machine learning research.

This topic covers setup instructions and the necessary files to build, test, and run JAX with ROCm support in a Docker environment. To learn more about JAX on ROCm, including its use cases, recommendations, as well as hardware and software compatibility, see JAX compatibility.

7.1 Install JAX on ROCm

To install JAX on ROCm, you have the following options:

- Use a prebuilt Docker image with JAX preinstalled (recommended)
- Use a ROCm base Docker image to install JAX
- Install JAX on bare-metal or a custom container
- Build JAX from source

7.1.1 Use a prebuilt Docker image with JAX preinstalled

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:latest
```

> **Note**
>
> For specific versions of JAX, review the periodically pushed Docker images at ROCm JAX on Docker Hub.

2. Once the image is downloaded, launch a container using the following command:

```
docker run -it \
    --network=host \
    --device=/dev/kfd \
    --device=/dev/dri \
    --ipc=host \
    --shm-size 64G \
    --group-add video \
```

(continues on next page)

(continued from previous page)

```
--cap-add=SYS_PTRACE \
--security-opt seccomp=unconfined \
-v $(pwd):/jax_dir \
--name rocm_jax \
rocm/jax:latest /bin/bash
```

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.

3. Verify the installation of ROCm JAX. See Test the JAX installation.

7.1.2 Docker image support

AMD validates and publishes ready-made JAX images with ROCm backends on Docker Hub. The following Docker image tags and associated inventories are validated for ROCm 7.2.3. For jax-community images, see rocm/jax-community on Docker Hub.

JAX 0.8.2

Python 3.12

Docker pull tag

```
docker pull rocm/jax:rocm7.2.3-jax0.8.2-py3.12
```

See rocm/jax:rocm7.2.3-jax0.8.2-py3.12 on Docker Hub.

Python 3.11

Docker pull tag

```
docker pull rocm/jax:rocm7.2.3-jax0.8.2-py3.11
```

See rocm/jax:rocm7.2.3-jax0.8.2-py3.11 on Docker Hub.

7.1.3 Use a ROCm base Docker image to install 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-24.04:7.2.3-complete
   ```

2. Launch the Docker container. After pulling the image, launch a container using this command:

```
docker run -it \
    --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-24.04:7.2.3-complete /bin/bash
```

3. Install the latest version of JAX. Inside the running container, install the required version of JAX with ROCm support using pip:

```
pip3 install --break-system-packages jax==0.8.2
pip3 install --break-system-packages jax-rocm7-pjrt==0.8.2
pip3 install --break-system-packages jax-rocm7-plugin==0.8.2
pip3 install --break-system-packages https://github.com/ROCm/rocm-jax/releases/download/rocm-
↪jax-v0.8.2/jaxlib-0.8.2+rocm7-cp312-cp312-manylinux_2_27_x86_64.manylinux_2_28_x86_64.
↪whl
```

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==0.8.2
jax-rocm7-pjrt==0.8.2
jax-rocm7-plugin==0.8.2
jaxlib==0.8.2
```

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. Install libdw1 if needed

```
apt update
apt install libdw1
```

7. Verify the installation of ROCm JAX. See Test the JAX installation.

7.1.4 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:

```
amd-smi
```

```
+------------------------------------------------------------------------------+
| AMD-SMI 26.2.1+fc0010cf6a    amdgpu version: 6.14.14  ROCm version: 7.2.3    |
| VBIOS version: 023.040.001.008.000001                                        |
| Platform: Linux Baremetal                                                    |
|-------------------------------------+----------------------------------------|
| BDF                        GPU-Name | Mem-Uti   Temp   UEC       Power-Usage |
| GPU  HIP-ID  OAM-ID  Partition-Mode | GFX-Uti    Fan               Mem-Usage |
|=====================================+========================================
| 0000:05:00.0    AMD Instinct MI355X | 0 %      54 °C   0         234/1400 W |
|   0       1       6        SPX/NPS1 | 0 %         N/A            283/294896 MB |
|-------------------------------------+----------------------------------------|
| 0000:15:00.0    AMD Instinct MI355X | 0 %      54 °C   0         238/1400 W |
|   1       3       7        SPX/NPS1 | 0 %         N/A            283/294896 MB |
+-------------------------------------+----------------------------------------+
+------------------------------------------------------------------------------+
| Processes:                                                                   |
|  GPU        PID  Process Name          GTT_MEM  VRAM_MEM  MEM_USAGE     CU % |
|==============================================================================
|  No running processes found                                                  |
+------------------------------------------------------------------------------+
```

2. Install the required version of JAX with ROCm support using pip:

```
pip3 install --break-system-packages jax==0.8.2
pip3 install --break-system-packages jax-rocm7-pjrt==0.8.2
pip3 install --break-system-packages jax-rocm7-plugin==0.8.2
pip3 install --break-system-packages https://github.com/ROCm/rocm-jax/releases/download/rocm-
↪jax-v0.8.2/jaxlib-0.8.2+rocm7-cp312-cp312-manylinux_2_27_x86_64.manylinux_2_28_x86_64.
↪whl
```

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. Install libdw1 if needed

```
apt update
apt install libdw1
```

6. Verify the installation of ROCm JAX. See Test the JAX installation.

7.1.5 Build JAX from source

The https://github.com/ROCm/rocm-jax/tree/rocm-jaxlib-v0.8.2 repository contains sources for the ROCm plugin for JAX as well as Dockerfiles used to build the AMD rocm/jax images. For the most up-to-date instructions, refer directly to the instructions in the repository:

- See Quick build for concise high-level steps.

- See Building for more in-depth build instructions and troubleshooting suggestions.

7.2 Test the JAX installation

After launching the container, test whether JAX detects ROCm devices as expected:

```
python3 -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

EIGHT

DGL ON ROCM INSTALLATION

Deep Graph Library (DGL) is an easy-to-use, high-performance, and scalable Python package for deep learning on graphs.

This topic covers setup instructions and the necessary files to build, test, and run DGL with ROCm support in a Docker environment. To learn more about DGL on ROCm, including its use cases, recommendations, as well as hardware and software compatibility, see DGL compatibility.

> **Note**
>
> DGL is supported on ROCm 7.0.0, 6.4.3, and 6.4.0. This topic provides installation instructions for ROCm 7.0.0 and 6.4.3. For ROCm 6.4.0, see previous-versions/dgl-history.

8.1 Install DGL

To install DGL on ROCm, you have the following options:

- Use the prebuilt Docker image (recommended)
- Build your own docker image
- Use a wheels package

8.1.1 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-2.4.0.amd0_rocm7.0.0_ubuntu24.04_py3.12_pytorch_2.8.0.
>
> 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

```
sudo docker run -it --network=host --device=/dev/kfd --device=/dev/dri \
--group-add=video --ipc=host --cap-add=SYS_PTRACE --security-opt \
seccomp=unconfined --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.

8.1.2 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 their respective ROCm versions listed below.

ROCm 7.0.0

PyTorch 2.8.0

Tag
: rocm/dgl:dgl-2.4.0.amd0_rocm7.0.0_ubuntu24.04_py3.12_pytorch_2.8.0

Inventory

- ROCm 7.0.0
- Python 3.12.9
- PyTorch 2.8.0

PyTorch 2.6.0

Tag
: rocm/dgl:dgl-2.4.0.amd0_rocm7.0.0_ubuntu24.04_py3.12_pytorch_2.6.0

Inventory

- ROCm 7.0.0
- Python 3.12.9
- PyTorch 2.6.0

PyTorch 2.7.1

Tag
: rocm/dgl:dgl-2.4.0.amd0_rocm7.0.0_ubuntu22.04_py3.10_pytorch_2.7.1

Inventory

- ROCm 7.0.0
- Python 3.10.16
- PyTorch 2.7.1

ROCm 6.4.3

PyTorch 2.6.0

Tag
: rocm/dgl:dgl-2.4.0.amd0_rocm6.4.3_ubuntu24.04_py3.12_pytorch_2.6.0

Inventory

- ROCm 6.4.3
- Python 3.12.9
- PyTorch 2.6.0

8.1.3 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 24.04 + ROCm 7.0.0 + Py 3.12 + PyTorch 2.8.0

To build the Docker container, run the following command:

```
# DGL on Ubuntu 24.04 + ROCm 7.0.0 + Py 3.12 + PyTorch 2.8.0
docker build \
  -t dgl:dgl-2.4.0.amd0_rocm7.0.0_ubuntu24.04_py3.12_pytorch_2.8.0 \
  --build-arg BASE_IMAGE=rocm/pytorch:rocm7.0_ubuntu24.04_py3.12_pytorch_release_2.8.0␣
↪\
  --build-arg ARG_MAX_JOBS=8 \
  --build-arg ARG_GPU_BUILD_TARGETS="gfx90a,gfx942" \
  --build-arg ARG_DGL_ARTIFACTS_DIR="/artifacts" \
  -f docker/Dockerfile.ci_gpu_rocm \
  .
```

DGL on Ubuntu 24.04 + ROCm 7.0.0 + Py 3.12 + PyTorch 2.6.0

To build the Docker container, run the following command:

```
# DGL on Ubuntu 24.04 + ROCm 7.0 + Py 3.12 + PyTorch 2.6.0
docker build \
  -t dgl:dgl-2.4.0.amd0_rocm7.0.0_ubuntu24.04_py3.12_pytorch_2.6.0 \
  --build-arg BASE_IMAGE=rocm/pytorch:rocm7.0_ubuntu24.04_py3.12_pytorch_release_2.6.0␣
↪\
  --build-arg ARG_MAX_JOBS=8 \
  --build-arg ARG_GPU_BUILD_TARGETS="gfx90a,gfx942" \
  --build-arg ARG_DGL_ARTIFACTS_DIR="/artifacts" \
  -f docker/Dockerfile.ci_gpu_rocm \
  .
```

DGL on Ubuntu 22.04 + ROCm 7.0.0 + Py 3.10 + PyTorch 2.7.1

To build the Docker container, run the following command:

```
# DGL on Ubuntu 22.04 + ROCm 7.0.0 + Py 3.10 + PyTorch 2.7.1
docker build \
  -t dgl:dgl-2.4.0.amd0_rocm7.0.0_ubuntu22.04_py3.10_pytorch_2.7.1 \
  --build-arg BASE_IMAGE=rocm/pytorch:rocm7.0_ubuntu22.04_py3.10_pytorch_release_2.7.1␣
↪\
  --build-arg ARG_MAX_JOBS=8 \
  --build-arg ARG_GPU_BUILD_TARGETS="gfx90a,gfx942" \
  --build-arg ARG_DGL_ARTIFACTS_DIR="/artifacts" \
  -f docker/Dockerfile.ci_gpu_rocm \
  .
```

DGL on Ubuntu 24.04 + ROCm 6.4.3 + Py 3.12 + PyTorch 2.6.0

To build the Docker container, run the following command:

```
# DGL on Ubuntu 24.04 + ROCm 6.4.3 + Py 3.12 + PyTorch 2.6.0
docker build \
  -t dgl:dgl-2.4.0.amd0_rocm6.4.3_ubuntu24.04_py3.12_pytorch_2.6.0 \
  --build-arg BASE_IMAGE=rocm/pytorch:rocm6.4.3_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 \
  .
```

8.1.4 Use a wheels package

The DGL .whl packages are hosted on the AMD PyPI repository. Instead of manually downloading the files, you can simply install DGL using pip with the provided URL. This command will automatically download and install the appropriate .whl file.

DGL on Ubuntu 24.04 + ROCm 7.0.0 + Py 3.12 + PyTorch 2.8.0

To install using wheels, run the following command:

```
pip install https://pypi.amd.com/rocm-7.0.0/packages/amd-dgl/amd_dgl-2.4.0+amd0.torch2.8.0.rocm7.0.
↪0.git64359f59.ubuntu24.4-cp312-cp312-linux_x86_64.whl
```

DGL on Ubuntu 24.04 + ROCm 7.0.0 + Py 3.12 + PyTorch 2.6.0

To install using wheels, run the following command:

```
pip install https://pypi.amd.com/rocm-7.0.0/packages/amd-dgl/amd_dgl-2.4.0+amd0.torch2.6.0.rocm7.0.
↪0.git2e48b21f.ubuntu24.4-cp312-cp312-linux_x86_64.whl
```

DGL on Ubuntu 22.04 + ROCm 7.0.0 + Py 3.10 + PyTorch 2.7.1

To install using wheels, run the following command:

```
pip install https://pypi.amd.com/rocm-7.0.0/packages/amd-dgl/amd_dgl-2.4.0+amd0.torch2.7.1.rocm7.0.
↪0.git698b58a9.ubuntu22.4-cp310-cp310-linux_x86_64.whl
```

DGL on Ubuntu 24.04 + ROCm 6.4.3 + Py 3.12 + PyTorch 2.6.0

To install using wheels, run the following command:

```
pip install https://pypi.amd.com/rocm-6.4.3/packages/amd-dgl/amd_dgl-2.4.0+amd0.torch2.6.0.
↪gitdbfe118.ubuntu24.4-cp312-cp312-linux_x86_64.whl
```

After installing a .whl file, you can confirm that the package was installed successfully using:

```
pip show amd_dgl
```

8.2 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"
```

8.2.1 Run tests for DGL

You can also run tests from the scripts folder at /src/dgl/. These tests are only viable if you are using the above Docker containers or building Docker containers from source.

To run Python tests:

```
cd /src/dgl
bash tests/scripts/task_unit_test_rocm.sh pytorch gpu
```

To run C++ tests:

```
cd /src/dgl
bash tests/scripts/task_cpp_unit_test.sh
```

Estimated time: the tests usually take 10-15 minutes to run.

8.3 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. For use cases and recommendations, refer to the AMD ROCm blog, where you can search for DGL examples and best practices to optimize your workloads on AMD GPUs.

8.4 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.

8.5 Previous versions

See previous-versions/dgl-history to find documentation for previous releases of the ROCm/dgl Docker image.

CHAPTER

NINE

RUNNING ROCM DOCKER CONTAINERS

Using Docker to run your ROCm applications is one of the best ways to get consistent and reproducible environments.

9.1 Prerequisites

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 ROCm installation instructions (which comes with amdgpu-dkms) or install amdgpu-dkms separately.

> **See also**
>
> For instructions on installing Docker, see the official Docker installation documentation.

9.2 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.

9.2.1 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.

9.2.2 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 ..
```

Note

When GPUs are partitioned (such as the Instinct MI300X or MI350X Series in DPX, QPX, or CPX mode), you must account for the number of partitions when selecting GPUs. For example, in CPX mode, renderD128 and renderD137 correspond to the first and second GPUs. In CPX mode, renderD128 to renderD136 correspond to different partitions of the first GPU. For more information, see GPU partition.

9.2.3 Verifying the AMD GPU 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.

9.3 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 and ROCm versions, and are a great starting place for building applications.

9.3.1 Pull a ROCm dev Docker image

Pull a ROCm dev Docker image with a supported configuration. See Docker Hub to browse available images. For example:

ROCm 7.1.1

Ubuntu 24.04

```
docker pull rocm/dev-ubuntu-24.04:7.1.1-complete
```

See rocm/dev-ubuntu-24.04:7.1.1-complete on Docker Hub.

Ubuntu 22.04

```
docker pull rocm/dev-ubuntu-22.04:7.1.1-complete
```

See rocm/dev-ubuntu-22.04:7.1.1-complete on Docker Hub.

9.3.2 Launch the Docker container

Launch the Docker container to allow GPU access. For example:

ROCm 7.1.1

Ubuntu 24.04

```
docker run -it \
    --cap-add=SYS_PTRACE \
    --ipc=host \
    --privileged=true \
    --shm-size=128GB \
    --network=host \
    --device=/dev/kfd \
    --device=/dev/dri \
    --group-add video \
    -v $HOME:$HOME \
    --name rocm7 \
    rocm/dev-ubuntu-24.04:7.1.1-complete
```

Ubuntu 22.04

```
docker run -it \
    --cap-add=SYS_PTRACE \
    --ipc=host \
    --privileged=true \
    --shm-size=128GB \
    --network=host \
    --device=/dev/kfd \
    --device=/dev/dri \
    --group-add video \
    -v $HOME:$HOME \
    --name rocm7 \
    rocm/dev-ubuntu-22.04:7.1.1-complete
```

9.3.3 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

TEN

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.

10.1 Installing prerequisites for Spack

> **Note**
>
> You must install all prerequisites before installing Spack.

Ubuntu

```
# Install some essential utilities:
apt-get -y update
apt-get -y install make patch bash tar gzip unzip bzip2 file gnupg2 git gawk
apt-get -y update
apt-get -y install xz-utils
apt-get -y install build-essential
apt-get -y install vim
apt-get -y install libpci-dev
# Install Python:
apt-get -y install python3
apt-get -y upgrade python3-pip
# Install Compilers:
apt-get -y install gcc
apt-get -y install gfortran
apt-get -y install liblzma-dev
apt-get -y install libbz2-dev
```

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++
```

10.2 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
   ```

After running setup-env.sh, the develop branch of the Spack packages repository will be cloned and used.

> **Note**
>
> To use your own local version of spack packages execute the following command:
>
> ```
> spack repo set --destination /path/to/local/spack-packages builtin
> ```

10.3 ROCm packages in Spack

> **Note**
>
> The supported ROCm components and their versions listed below were accurate as of the time of initial ROCm release. For the most up-to-date information, see the latest version of this information at ROCm packages in Spack.

Component	Spack package name	Minimum supported version	Latest supported version
AMD SMI	amdsmi	5.6.0	7.2.1
aqlprofile	hsa-amd-aqlprofile	7.0.0	7.2.1
comgr	comgr	5.6.0	7.2.1
Composable Kernel	composable-kernel	5.6.0	7.2.1
devicelibs	rocm-device-libs	5.6.0	7.2.1
HIP (hip_in_vdi)	hip	5.6.0	7.2.1
hipBLAS	hipblas	5.6.0	7.2.1
hipBLAS-common	hipblas-common	6.3.0	7.2.1
hipBLASLt	hipblaslt	6.0.0	7.2.1
HIPCC	hipcc	5.7.0	7.2.1
hipCUB	hipcub	5.6.0	7.2.1
hipFFT	hipfft	5.6.0	7.2.1
hipfort	hipfort	5.6.0	7.2.1
HIPIFY	hipify-clang	5.6.0	7.2.1
hipRAND	hiprand	5.6.0	7.2.1
hipSOLVER	hipsolver	5.6.0	7.2.1
hipSPARSE	hipsparse	5.6.0	7.2.1
hipSPARSELt	hipsparselt	6.0.0	7.2.1
hipTensor	hip-tensor	5.7.0	7.2.1
HIP Tests	hip-tests	6.1.0	7.2.1
llvm	llvm-amdgpu	5.6.0	7.2.1
MIGraphX	migraphx	5.6.0	7.2.1
MIOpen (HIP)	miopen-hip	5.6.0	7.2.1
MIVisionX	mivisionx	5.6.0	7.2.1
OpenCL	rocm-opencl	5.6.0	7.2.1
openmp-extras	rocm-openmp-extras	5.6.0	7.2.1
RCCL	rccl	5.6.0	7.2.1
rocAL	rocal	6.2.0	7.2.1
rocALUTION	rocalution	5.6.0	7.2.1
rocBLAS	rocblas	5.6.0	7.2.1
ROCdbgapi	rocm-dbgapi	5.6.0	7.2.1
rocDecode	rocdecode	6.1.0	7.2.1
rocFFT	rocfft	5.6.0	7.2.1
rocJPEG	rocjpeg	6.3.0	7.2.1
rocm-core	rocm-core	5.6.0	7.2.1
rocminfo	rocminfo	5.6.0	7.2.1
rocMLIR	rocmlir	5.4.0	7.2.1
ROCm Bandwidth Test	rocm-bandwidth-test	5.6.0	7.2.1
ROCm CMake	rocm-cmake	5.6.0	7.2.1
ROCm Compute Profiler	rocprofiler-compute	6.3.2	7.2.1
ROCm Data Center Tool (RDC)	rdc	5.6.0	7.2.1
ROCm Debug Agent	rocm-debug-agent	5.6.0	7.2.1
ROCm Debugger (ROCgdb)	rocm-gdb	5.6.0	7.2.1
ROCm Examples	rocm-examples	6.2.0	7.2.1
ROCm SMI Library	rocm-smi-lib	5.6.0	7.2.1
ROCm Systems Profiler	rocprofiler-systems	6.3.0	7.2.1
ROCm Validation Suite	rocm-validation-suite	5.6.0	7.2.1
rocPRIM	rocprim	5.6.0	7.2.1
ROCProfiler	rocprofiler-dev	5.6.0	7.2.1
rocprofiler-register	rocprofiler-register	6.1.0	7.2.1
ROCprofiler-SDK	rocprofiler-sdk	6.2.4	7.2.1

continues on next page

Table 1 – continued from previous page

Component	Spack package name	Minimum supported version	Latest supported version
rocPyDecode	rocpydecode	6.2.0	7.2.1
rocRAND	rocrand	5.6.0	7.2.1
ROCr Runtime	hsa-rocr-dev	5.6.0	7.2.1
rocSHMEM	rocshmem	6.4.0	7.2.1
rocSOLVER	rocsolver	5.6.0	7.2.1
rocSPARSE	rocsparse	5.6.0	7.2.1
rocThrust	rocthrust	5.6.0	7.2.1
ROCTracer	roctracer-dev	5.6.0	7.2.1
roctracer-dev-api	roctracer-dev-api	5.6.0	7.2.1
rocWMMA	rocwmma	5.6.0	7.2.1
ROCm Performance Primitives (RPP)	rpp	5.7.0	7.2.1
TransferBench	transferbench	6.3.0	7.2.1
aqlprofile (old)	aqlprofile	5.6.0	6.4.3 (final)
clang-ocl	rocm-clang-ocl	5.6.0	6.1.2 (final)
Omniperf	omniperf	6.2.0	6.3.1 (final)
Omnitrace	omnitrace	rocm-6.2.0	rocm-6.3.0 (final)
ROCT Thunk Interface	hsakmt-roct	5.6.0	6.2.4 (final)
Tensile	rocm-tensile	5.6.0	7.2.1

10.4 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@7.2.1

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
   ```

(continues on next page)

(continued from previous page)

```
Preferred version:
    7.2.1    https://github.com/ROCm/MIVisionX/archive/rocm-7.2.1.tar.gz

Safe versions:
    7.2.1    https://github.com/ROCm/MIVisionX/archive/rocm-7.2.1.tar.gz
    7.2.0    https://github.com/ROCm/MIVisionX/archive/rocm-7.2.0.tar.gz
    7.1.1    https://github.com/ROCm/MIVisionX/archive/rocm-7.1.1.tar.gz
    7.1.0    https://github.com/ROCm/MIVisionX/archive/rocm-7.1.0.tar.gz
    7.0.2    https://github.com/ROCm/MIVisionX/archive/rocm-7.0.2.tar.gz
    7.0.0    https://github.com/ROCm/MIVisionX/archive/rocm-7.0.0.tar.gz
    6.4.3    https://github.com/ROCm/MIVisionX/archive/rocm-6.4.3.tar.gz
    6.4.3    https://github.com/ROCm/MIVisionX/archive/rocm-6.4.3.tar.gz
    6.4.2    https://github.com/ROCm/MIVisionX/archive/rocm-6.4.2.tar.gz
    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
    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

Deprecated versions:
    None

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

    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
```

(continues on next page)

(continued from previous page)

```
Build Dependencies:
    cmake ffmpeg hip libjpeg-turbo lmdb miopen-hip opencv protobuf py-google-
↪api-python-client py-protobuf py-pytz py-wheel rapidjson rpp
    cxx gmake hsa-rocr-dev llvm-amdgpu migraphx ninja openssl py-future py-numpy ␣
↪ py-pybind11 py-setuptools python rocm-core

Link Dependencies:
    hip hsa-rocr-dev llvm-amdgpu lmdb migraphx miopen-hip openssl py-future py-google-api-
↪python-client py-numpy py-pybind11 py-pytz py-setuptools py-wheel rapidjson rocm-core ␣
↪rpp

Run Dependencies:
    py-protobuf

Licenses:
    MIT
```

10.5 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@7.2.1%gcc@13.2.0~add_tests~asan+hip~ipo~opencl build_system=cmake build_
↪type=Release generator=make arch=linux-ubuntu24.04-skylake_avx512
-       ^cmake@3.28.3%gcc@13.2.0~doc+ncurses+ownlibs~qtgui build_system=generic build_
↪type=Release patches=dbc3892 arch=linux-ubuntu24.04-skylake_avx512
-           ^ffmpeg@4.4.4%gcc@13.2.0~X~avresample+bzlib~doc~drawtext+gpl~libaom~libmp3lame~
↪libopenjpeg~libopus~libsnappy~libspeex~libssh~libvorbis~libvpx~libwebp~libx264~libxml2~libzmq~
↪lzma~nonfree~openssl~sdl2+shared+version3 build_system=autotools patches=f070ac1␣
↪arch=linux-ubuntu24.04-skylake_avx512
-               ^alsa-lib@1.2.3.2%gcc@13.2.0~python build_system=autotools arch=linux-ubuntu24.04-
↪skylake_avx512
-               ^bzip2@1.0.8%gcc@13.2.0~debug~pic+shared build_system=generic arch=linux-
↪ubuntu24.04-skylake_avx512
-               ^libiconv@1.17%gcc@13.2.0 build_system=autotools libs=shared,static arch=linux-
↪ubuntu24.04-skylake_avx512
...
```

10.6 Creating an environment

You can create an environment with all the required components of your version, install them collectively, and work in the environment.

1. Create a Spack environment.

```
spack env create myenv
```

2. Activate the created environment.

```
spack env activate myenv
```

3. Add the ROCm packages.

```
spack add
rocm-cmake@7.2.1 rocm-dbgapi@7.2.1 rocm-debug-agent@7.2.1 rocm-gdb@7.2.1 rocminfo@7.2.1 \
rocm-opencl@7.2.1 rocm-smi-lib@7.2.1 rocprim@7.2.1 rocprofiler-dev@7.2.1 rocrand@7.2.1  \
rocthrust@7.2.1 roctracer-dev@7.2.1
```

4. Generate the build plan.

```
spack concretize
```

5. Install the packages.

```
spack install
```

10.7 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@7.2.1
# (This will pull the 7.2.1 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-7.2.1/spack/var/spack/cache/_source-cache/
↪archive/d8/d8dba8cdf05463afb7879de2833983cafa6a006ba719815a35b96d9b92fc7fc4.tar.gz
==> Using cached archive: /data/root/temp/rocm-7.2.1/spack/var/spack/cache/_source-cache/
↪archive/82/829e61a5c54d0c8325d02b0191c0c8254b5740e63b8bfdb05eec9e03d48f7d2c.tar.gz
==> Using cached archive: /data/root/temp/rocm-7.2.1/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-7.2.1-zo53ondw3tevsr3gmoofbhre7asvis46/
↪spack-src/clr
==> Moving resource stage
```

(continues on next page)

(continued from previous page)

```
    source: /tmp/root/spack-stage/resource-hip-tests-zo53ondw3tevsr3gmoofbhre7asvis46/spack-
↪src/
    destination: /tmp/root/spack-stage/spack-stage-hip-7.2.1-zo53ondw3tevsr3gmoofbhre7asvis46/
↪spack-src/hip-tests
==> Staged hip in /tmp/root/spack-stage/spack-stage-hip-7.2.1-zo53ondw3tevsr3gmoofbhre7asvis46
```

2. Change directory to spack-src inside the staged directory.

```
/spack$ cd /tmp/root/spack-stage/spack-stage-hip-7.2.1-zo53ondw3tevsr3gmoofbhre7asvis46
/tmp/root/spack-stage/spack-stage-hip-7.2.1-zo53ondw3tevsr3gmoofbhre7asvis46$ cd spack-src/
```

3. Create a new Git repository.

```
/tmp/root/spack-stage/spack-stage-hip-7.2.1-zo53ondw3tevsr3gmoofbhre7asvis46/spack-src$ git init
```

4. Add the entire directory to the repository.

```
/tmp/root/spack-stage/spack-stage-hip-7.2.1-zo53ondw3tevsr3gmoofbhre7asvis46/spack-src$ git␣
↪add .
```

5. Make the required changes to the source code.

```
/tmp/root/spack-stage/spack-stage-hip-7.2.1-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
/tmp/root/spack-stage/spack-stage-hip-7.2.1-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.

```
/tmp/root/spack-stage/spack-stage-hip-7.2.1-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="@7.2.1")
```

   Spack will apply 0001-modifications.patch on the 7.2.1 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

ELEVEN

PACKAGE DETAILS

This section provides information about the required meta-packages for the following AMD ROCm programming models:

- Heterogeneous-Computing Interface for Portability (HIP)
- OpenCL™
- OpenMP™

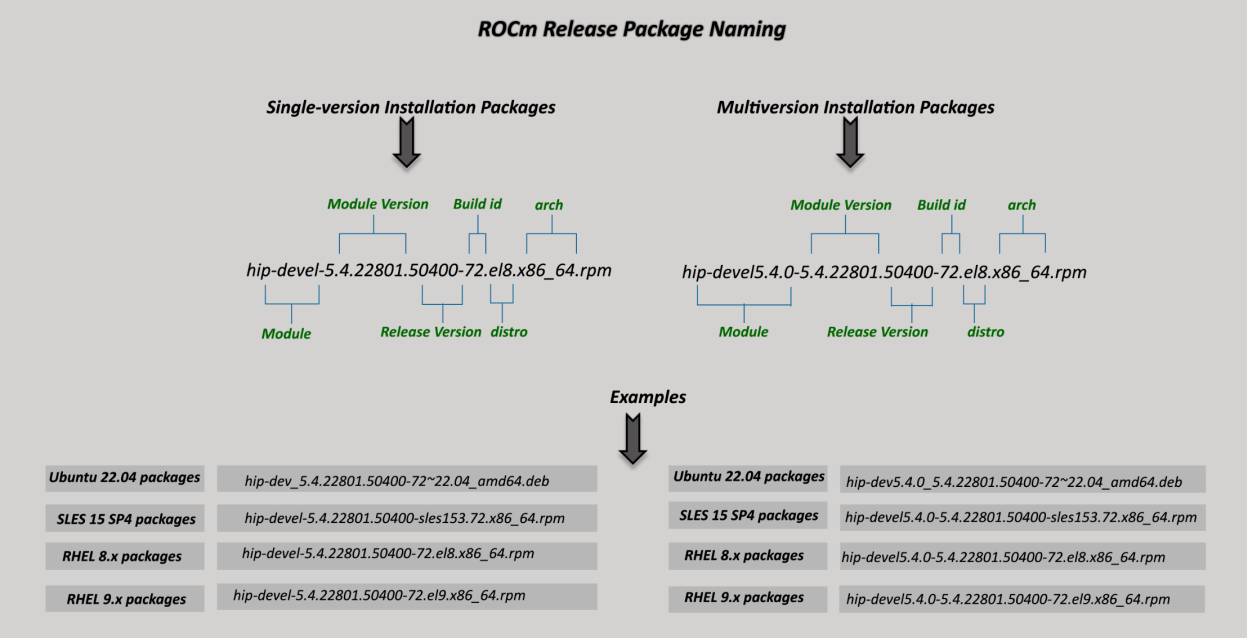
11.1 ROCm package naming conventions

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.

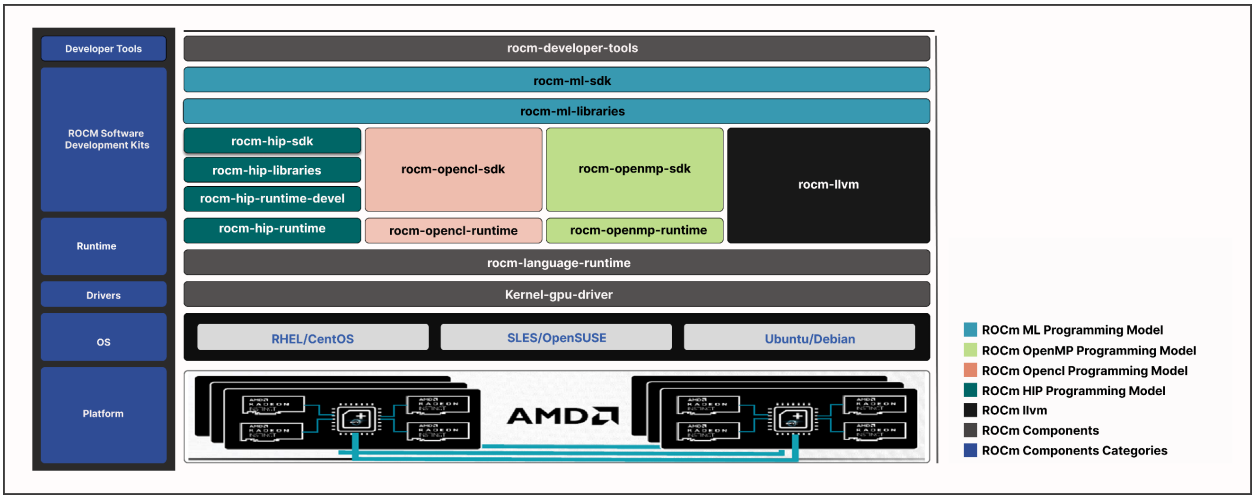
Arch - It shows the architecture for which the package was created.

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.

11.2 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.

ROCm installation can be tailored to your requirements using one more combinations of ROCm meta packages:

- To use pre-built ROCm libraries and tools, include ROCm runtime packages in the installation step.
- To develop and build individual ROCm libraries and tools, include ROCm developer packages in the installation step.

11.2.1 ROCm runtime packages

Meta package	Description	Legacy use case[1]
rocm	All ROCm core packages, tools, and libraries.	rocm
rocm-hip-libraries	HIP libraries optimized for the AMD platform.	Legacy use case does not exist.
rocm-hip-runtime	Run HIP applications written for the AMD platform.	hip
rocm-language-runti	ROCm runtime environment for running applications on the AMD platform.	lrt
rocm-ml-libraries	Key machine learning libraries. Includes MIOpen.	mllib
rocm-opencl-runtim	Run OpenCL-based applications on the AMD platform.	opencl
Other package		
amdgpu-lib	For users of graphics applications which require the open source Mesa 3D graphics and multimedia libraries. This package is primarily used for Radeon GPUs.	graphics

11.2.2 ROCm developer packages

Meta package	Description	Legacy use case[1]
rocm-developer-tool	Debug and profile HIP applications.	rocmdevtools
rocm-hip-runtime-d	Develop applications on HIP or port from CUDA.	Legacy use case does not exist.
rocm-hip-runtime-d		
rocm-hip-sdk[3]	Develop or port HIP applications and libraries for the AMD platform.	hiplibsdk
rocm-ml-sdk	Develop and run machine learning applications for AMD.	mlsdk
rocm-opencl-sdk	Develop OpenCL-based applications for the AMD platform.	openclsdk
rocm-openmp-sdk	Develop OpenMP-based applications for the AMD software.	openmpsdk

[1] Starting from ROCm 6.4.2, "Legacy use cases" in amdgpu-install are replaced by the equivalent meta package. In addition, the following amdgpu-install use cases: asan, rocmdev, multimedia, multimediasdk, amf, and workstation are deprecated.
[2] Use rocm-hip-runtime-dev for Debian/Ubuntu-based systems and rocm-hip-runtime-devel for RHEL/RPM-based systems.
[3] rocm-hip-sdk requires rocm-hip-runtime-devel.

11.3 Packages in ROCm programming models

The following tables show the meta-packages and their associated (meta-)packages in a ROCm programming model.

> **Note**
>
> Some meta packages and dependencies in the tables below have names ending with -dev, as they are based on Ubuntu. On RPM-based systems like RHEL, these packages use a different naming convention and typically end with -devel instead.

11.3.1 ROCm runtime packages

Meta package	Associated meta packages or packages
rocm	Meta packages: rocm-developer-tools, rocm-hip, rocm-openmp, rocm-opencl-sdk Packages: half, migraphx, migraphx-dev, miopen-hip, miopen-hip-dev, mivisionx, mivisionx-dev, rocm-cmake, rocm-core, rocminfo, rocm-llvm, rpp, rpp-dev
rocm-hip-libraries	Meta package: 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	Meta package: rocm-language-runtime Packages: hip-runtime-amd, rocm-core, rocminfo
rocm-language-runti	Packages: comgr, hsa-rocr, openmp-extras-runtime, rocm-core
rocm-ml-libraries	Meta package: rocm-hip-libraries Packages: half, miopen-hip, rocm-core, rocm-llvm
rocm-opencl-runtim	Meta package: rocm-language-runtime Packages: rocm-core, rocm-opencl

11.3.2 ROCm developer packages

Meta package	Associated meta packages or packages
rocm-developer-tool:	Packages: amd-smi-lib, comgr, hsa-amd-aqlprofile, hsa-rocr, openmp-extras-runtime, rocm-core, rocm-dbgapi, rocm-debug-agent, rocm-gdb, rocm-smi-lib, rocprofiler, rocprofiler-compute, rocprofiler-dev, rocprofiler-plugins, rocprofiler-register, rocprofiler-sdk, rocprofiler-sdk-rocpd, rocprofiler-sdk-roctx, rocprofiler-systems, roctracer, roctracer-dev
rocm-hip-runtime-de	Meta package: rocm-hip-runtime Packages: hipcc, hip-dev, hip-doc, hipify-clang, hip-samples, hsa-rocr-dev, rocm-cmake, rocm-core, rocm-device-libs, rocm-llvm
rocm-hip-sdk	Meta packages: rocm-hip-libraries, rocm-hip-runtime-dev Packages: composablekernel-dev, hipblas-common-dev, hipblas-dev, hipblaslt-dev, hipcub-dev, hipfft-dev, hipfort-dev, hiprand-dev, hipsolver-dev, hipsparse-dev, hipsparselt-dev, hiptensor-dev, rccl-dev, rocalution-dev, rocblas-dev, rocfft-dev, rocm-core, rocprim-dev, rocrand-dev, rocsolver-dev, rocsparse-dev, rocthrust-dev, rocwmma-dev
rocm-ml-sdk	Meta packages: rocm-hip, rocm-ml-libraries Packages: miopen-hip-dev, rocm-core
rocm-opencl-sdk	Packages: comgr, hsa-rocr, hsa-rocr-dev, rocm-core, rocm-llvm, rocm-opencl, rocm-opencl-dev
rocm-openmp-sdk	Meta package: rocm-language-runtime Packages: hsa-rocr-dev, openmp-extras-dev, rocm-core, rocm-device-libs, rocm-llvm

CHAPTER

TWELVE

INSTALLATION TROUBLESHOOTING

Troubleshooting describes issues that some users encounter when installing the ROCm tools or libraries.

12.1 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)

Refer to ROCm Issue #2422 for additional details.

12.2 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.

12.3 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.

12.4 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++-<gcc-version>-dev
```

For more information on how to determine the relevant gcc-version, refer to ROCm Issue #1843.

12.5 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.

12.6 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 gpg
```

Debian

```
apt update
apt install sudo wget gpg
```

Red Hat Enterprise Linux

```
dnf install sudo wget
```

Oracle Linux

```
dnf install sudo wget
```

SUSE Linux Enterprise Server

```
zypper install sudo wget SUSEConnect awk
```

Rocky Linux

```
dnf install sudo wget
```

After installing these packages, install ROCm using the Quick start installation guide in your Docker container.

12.7 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.

12.8 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.

12.9 Issue #9: Cannot access the AMD GPU 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. To learn how to add an account to these groups, see Configuring permissions for GPU access.

12.10 Issue #10: ROCm debugging tools might become unresponsive in SELinux-enabled distributions

Red Hat Enterprise Linux (RHEL) and related distributions automatically enable a security feature named Security-Enhanced Linux (SELinux) that may prevent ROCm debugging tools like ROCgdb, ROCdbgapi, and ROCR Debug Agent from working correctly.

The problem occurs when attempting to debug a program that contains code that runs on the GPU. The debugging session may become unresponsive while attempting to reach a breakpoint or doing instruction-stepping in device code. ROCgdb will still be responsive and accept interruption by pressing Control+C, but the breakpoint in device code won't be hit, and the instruction-stepping operation will not conclude.

The ROCR Debug Agent might also become unresponsive when attempting to capture data from a program that is running into queue errors, memory faults, and other triggering events.

As a workaround for this problem, either disable SELinux or configure it to use the permissive setting.

While ROCgdb or ROCR Debug Agent are being used, setting SELinux to permissive can be accomplished with the following command:

```
sudo setenforce 0
```

After the session is over, it can be switched back to enforcing mode:

```
sudo setenforce 1
```

Note

Changing the SELinux settings can have security implications. Ensure you review your system security settings before making any changes.