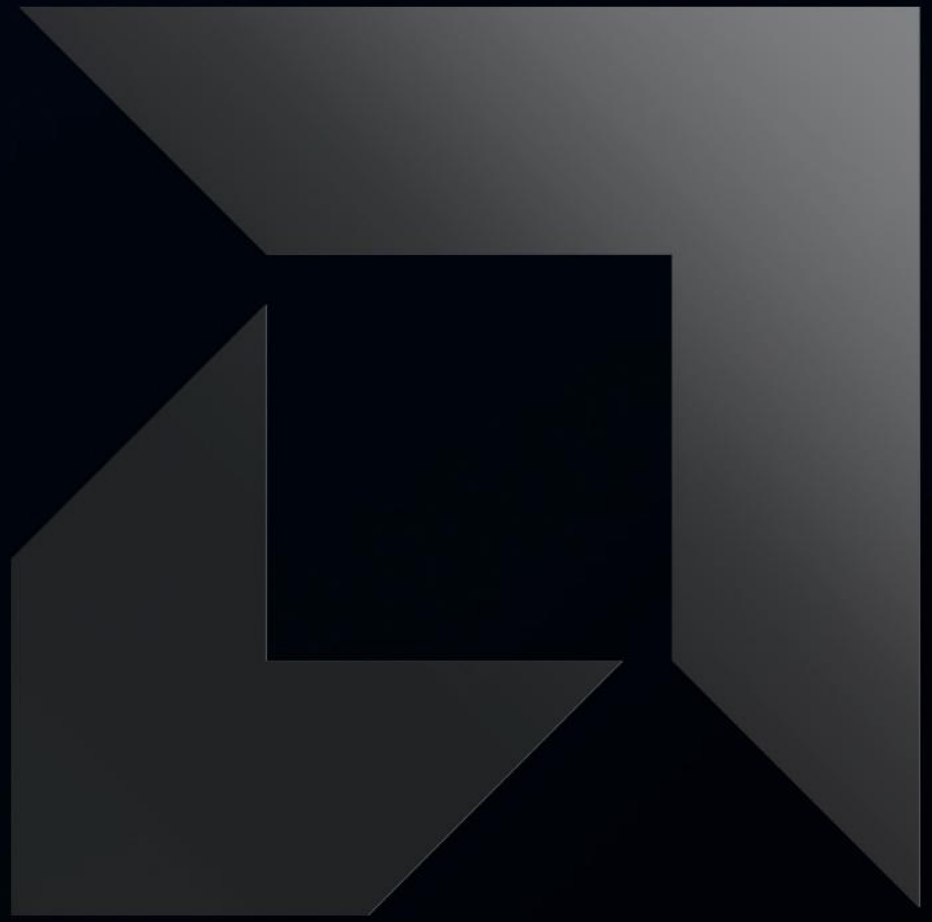


Tools and Resources for System Administration of AMD GPUs

Presenter: Bob Robey
AMD @ Tsukuba University
Oct 21-23, 2025

AMD
together we advance_

Why Sys Admin Tools and Resources: Use Cases

System Administration

- Reference for how to manually build wide-range of software for AMD GPU systems
- Recommended module setup
- Standardization of installations for better portability between sites
- Test relocating ROCm software to a different directory
- Pre-package software as a tarball to speed up and simplify installations
- Test various installation options such as an optimized build from source versus a generic pre-built download
- Cross-build packages on a system with a different GPU or no GPU at all
- Provide reproducible and easy way to setup environment for trainings/workshops

Application Development

- Reproducing bug reports
- Installing software on a local workstation
- Test software with different ROCm versions and Operating Systems, especially prior to an upgrade of a site
- Test software in a user-controlled environment

Introductory Remarks

While the **ROCm software install** is the core of the installation process, there are **additional installation steps** necessary to create a completely functioning system

- This presentation provides a high-level overview of the steps along with some of the key installation commands
- Automating the installation steps is recommended so future installs are easier
- The **model installation** shown in this presentation will offer an example of a **cluster setup**

WE WELCOME YOUR FEEDBACK AND CONTRIBUTIONS!

The model installation scripts that will be discussed in this presentation are continually evolving and improving and the set of distributions and software supported is also growing. We want to promote a crowd sourcing approach and welcome your feedback and contributions.

System Setup Roadmap

1. Installation design and planning
   - Overview
   - Container installation
   - Bare Metal installation
2. GUI interfaces for remote users
3. ROCm installation
4. AMD profiling tools and debuggers
5. Modules
6. Job schedulers
7. Communication libraries
8. HPC community tools
9. Additional packages
10. Testing the success of the installation
11. Final remarks

Installation design and planning

Overview

Presenter: Bob Robey
Organization: AMD

Determine the Purpose of the Cluster

Understanding the purpose of the cluster will help with decisions on what software to install

- Will there be application development or just running already existing software?
- Will there be multi-node application runs or just single node?
- Will this be a single- or multi-user system?
- Are you looking for stability or pushing the cutting edge?
- Are the applications mostly traditional HPC or Machine Learning/Artificial Intelligence?
- Will there be any special or parallel storage hardware?
- Are the users more comfortable with traditional HPC setups or prefer container environments?

Supported Hardware

Data Center GPUs, Workstation GPUs and Desktop GPUs are currently supported

- AMD Instinct™ MI300X, MI300A, MI250X, MI250, MI210, MI100
  - Necessary for multi-node scaling as well as more GPU muscle

- AMD Radeon Pro™ W6800, V620, VII
  - May give usable single GPU performance, but limited in memory

- AMD Radeon™ VII
  - More limited in memory

NOTE: others not listed **may work**, but have limited support

List of AMD GPUs in LLVM™ docs may help identify compiler support

- https://llvm.org/docs/AMDGPUUsage.html#processors

Supported Operating Systems

Three Linux® distributions are primarily supported

- Ubuntu® – popular for workstations and small clusters
- Red Hat® Enterprise Linux – used at larger HPC sites. Provides commercial support and specialized deployment tools. Also Rocky Linux and Oracle® Linux distros.
- SUSE® Linux Enterprise Server – an open-source distribution targeting large HPC and commercial sites
- Debian

Any of these three are excellent choices for the base distribution

A couple of the latest versions of each distribution are supported

- Ubuntu® – 22.04, 24.04 – these are the long-term support versions and updated every two years in April
- Redhat® –8.10, 9.4, 9.6
- SUSE® –15.7
- Debian – 12

➢ **We'll show instructions for Ubuntu 24.04**

Model Installation Resources

After deciding the purpose of the cluster, the following installation resources are available:

Model installation repository:

https://github.com/amd/HPCTrainingDock.git

- **Container – can build complete HPC node for testing before deployment**
- **Bare metal installation scripts**
  - Can test individual package installation scripts before being deployed
  - Installation testing process mimics a sys admin user
  - Script testing done in a container instead of the target system (e.g., can build containers on a CPU-only machine while targeting systems with GPUs)

  Check the **README.md** file for more details:
  https://github.com/amd/HPCTrainingDock/blob/main/README.md

Installation design and planning

Container installation

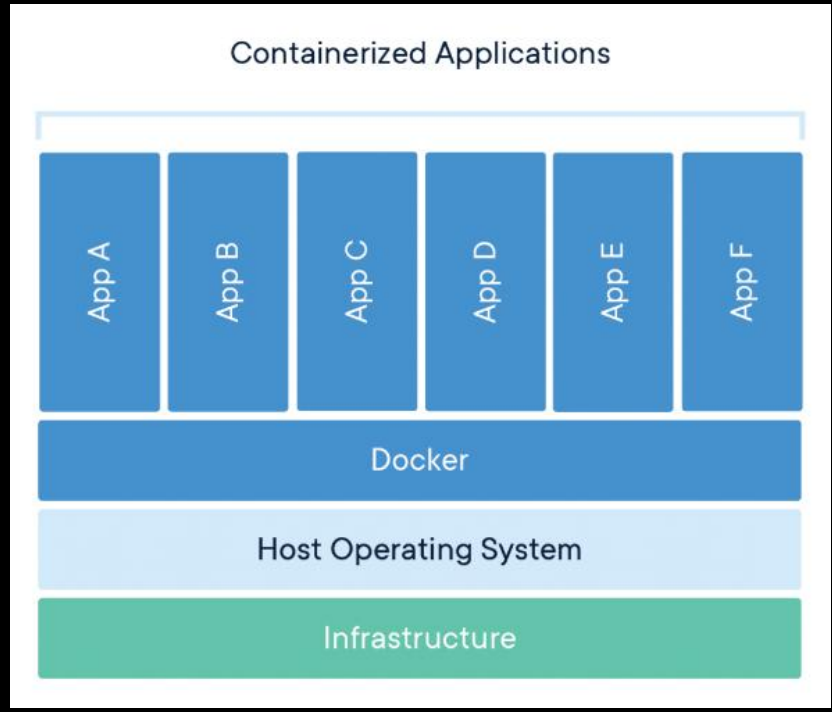
Presenter: Giacomo Capodaglio
Organization: AMD

Brief Review: What are Containers in this Context?

From the Docker® website: A container is a standard unit of software that packages up code and all its dependencies, so the application runs quickly and reliably from one computing environment to another.

Examples of containers:

- Docker®
  - Uses a daemon
  - Runs containers as `root` only
- Podman
  (Docker drop-in replacement)
  - Does not use a daemon
  - Runs containers both as `root` and as `non-root`
- Singularity
  - Similar features as Podman
  - Originally designed for HPC applications



(image source: Docker®)

Container Installation Instructions 1/2

Installation assumptions

- Docker® is installed
- Your `$USER` is part of the Docker® group
- You can issue Docker® commands without `sudo`

If **Podman** is installed instead of Docker, the installation scripts **will detect it** and automatically add

```
--format docker
```

to all the `docker build` commands

Our installation scripts can be also used to create a **Singularity** image. For details, please refer to the instructions in: https://github.com/amd/HPCTrainingDock/blob/main/README.md

Other container systems that follow the Open Container Initiative should work with some minor effort

Container Installation Instructions 2/2

The Docker® container is set up to use **Ubuntu® 22.04** as the Operating System

Will build four different images, called:

- rocm
- comm
- tools
- extras

To run the container setup script, use these commands from terminal:

```
git clone --recursive git@github.com:amd/HPCTrainingDock.git
cd HPCTrainingDock
 ./build-docker.sh --rocm-versions 6.3.2 --distro ubuntu --distro-versions 22.04
                   --admin-username <admin> --admin-password <password>
```

NOTE: a **password** has to be specified, whereas the `admin-username` is defaulted to admin

Container Installation Options

`--distro` : autodetected by looking into `/etc/os-release`

`--distro-versions` : autodetected by looking into `/etc/os-release`

`--rocm-versions` : default is 6.0

`--python-version` : this is the minor version of python3. Default is 10

`--docker-user` : default is the output of the `whoami` command

`--admin-username` : default is `admin`

`--admin-password` : there is **NO default**, and a password has to be specified as input argument

`--amdgpu-gfxmodel` : autodetected using `rocminfo`

`--install-omnitrace-research` : if included, it will install a pre-build version of Omnitrace research

`--omnitrace-build-from-source` : if included, it will install Omnitrace research from source

`--install-omniperf-research` : if included, it will install Omniperf research from source

`--push` : if included, it will push the image to Dockerhub

`--output-verbosity` : if included, it will add the `--progress=plain` option to the Docker® build

`--no-cache` : if included, it will add the `--no-cache` option to the Docker® build

`--no-pull` : if included, it will remove the option to pull from the Docker® build options for the ROCm image

`--retry` : default is 3

Container Optional Installation Options (off by default)

`--build-options`: use this to specify a semi-colon separated list of packages to build

`--build-aomp-latest` : AOMP is the version of Clang/LLVM™ with OpenMP® for AMD GPUs for future ROCm releases

`--build-llvm-latest` : build LLVM version from head of LLVM™ repository

`--build-gcc-latest` : GCC compiler with some offloading support, including Fortran

`--build-clacc-latest` : build branch of LLVM™ with conversion of OpenACC code built in for C applications

`--build-pytorch` : build PyTorch version with AMD GPU support

`--build-cupy` : build CuPy version with AMD GPU support

`--build-mpi4py` : build MPI4Py with AMD GPU support and using GPU-Aware OpenMPI

`--build-jax` : build JAX with AMD GPU support

`--build-hpctoolkit` : build HPCToolKit with AMD GPU support

`--build-tau` : build the TAU profiler with AMD GPU support

`--build-scorep` : build the Score-P performance with AMD GPU support

`--build-kokkos` : build Kokkos

`--build-x11vnc:` build x11vnc capabilities for screen forwarding through the container

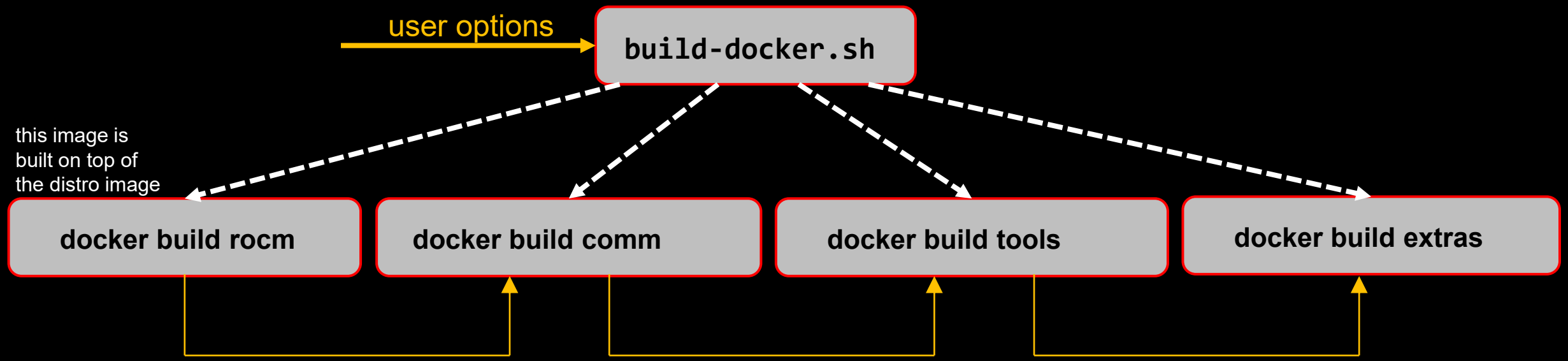
`--build-all-latest` : build aomp-latest, gcc-latest, pytorch, cupy, jax, tau, score-p, hpctoolkit, x11vnc and kokkos

`--use-cached-apps` : build cached apps instead of doing it from scratch

New packages are constantly being added. To see a full list of options, type: `./build-docker.sh --help`

NOTE: some packages such as ROCm and OpenMPI are always installed, and currently there are no build options to exclude them. If necessary, the call to their install script can be commented out from the Dockerfile

Container Installation Flow Chart



- baseospackages_setup.sh
- lmod_setup.sh
- rocm_setup.sh
- rocm_omnitrace_setup.sh
- rocm_omniperf_setup.sh
- slurm_setup.sh

- openmpi_setup.sh
- mvapich2_setup.sh
- mpi4py_setup.sh

- omnitrace_setup.sh
- scorep_setup.sh
- omniperf_setup.sh
- hpctoolkit_setup.sh
- tau_setup.sh
- grafana_setup.sh

- compiler_setup.sh
- amd_gcc_setup.sh
- aomp_setup.sh
- llvm-build.sh
- clacc_setup.sh
- kokkos_setup.sh
- cupy_setup.sh
- jax_setup.sh
- pytorch_setup.sh
- miniconda3_setup.sh
- x11vnc_setup.sh
- + more

NOTE: this flow chart only describes the **build process for the images**. Once the images have been built, it is necessary to run the container and then ssh into it to use it, as described in the next slides

Container Installation Instructions – Run

To run the container just do:

```
docker run -it --device=/dev/kfd  \
               --device=/dev/dri  \
               --group-add video render audio \
               -p 2222:22         \
               --detach           \
               --name Training    \
               --rm \
               -v $HOME/Class/training/hostdir:/hostdir \
               --security-opt seccomp=unconfined \
               docker.io/library/training
```

enable AMD GPUs on container

port number – map external port to internal port

run container in the background

remove container if it already exists

map host directory to container

sets secure computing mode without any of the Docker® default seccomp profiles

image

Then you can access it by doing: `ssh <admin_username>@localhost -p 2222`

and enter the password `<admin_password>` set at build time

Installation design and planning

Bare metal installation

Presenter: Giacomo Capodaglio
Organization: AMD

Bare System Installation Instructions 1/2

A set of scripts is available in the **model installation repo** to configure, install and test software packages for AMD GPUs

The bare system installation tests the installs as a regular sys admin user with sudo privileges
- This mimics the recommended way installs are done at most sites

The scripts are also called by Docker® during the container installation

Two main scripts are used to manage and organize the bare system install:

`main_setup.sh` : the main script that one-by-one calls the scripts to install each piece of software

`test_install.sh` : this script builds and runs a docker container to test the installation process initiated by `main_setup.sh`. It automatically gets you in the container, with user ID `sysadmid`

Bare System Installation Instructions 2/2

To run the scripts introduced in the previous slide, follow this procedure:

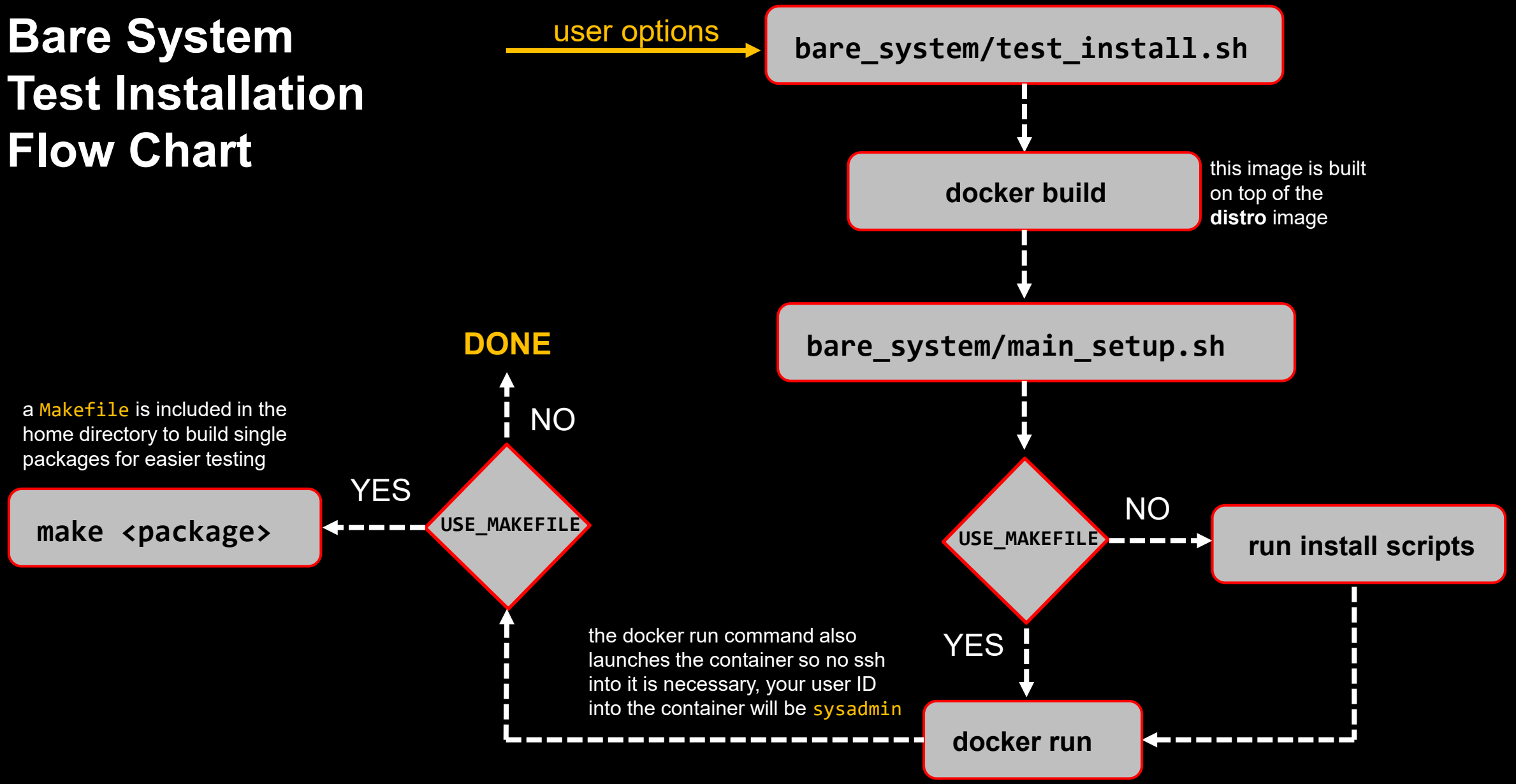
```
git clone --recursive git@github.com:amd/HPCTrainingDock.git && \
cd HPCTrainingDock && \
./bare_system/main_setup.sh +options
```

```
git clone --recursive git@github.com:amd/HPCTrainingDock.git && \
cd HPCTrainingDock && \
./bare_system/test_install.sh +options
```

NOTE: these scripts have to be run from the `HPCTrainingDock` directory, NOT from the `bare_system` directory

The options you can provide will be discussed in the next slides

Bare System Test Installation Flow Chart



Bare System Installation Instructions – Options

Options for `main_setup.sh`:

`--rocm-version` : default is 6.0

`--rocm-install-path` : default is `/opt/`

`--python-version` : minor versions for python3, default is 10

`--amdgpu-gfxmodel` : no default but `rocminfo` is used to assign a value to it, if a value is not provided

`--install-omnitrace-research` : default is *false*

`--install-omniperf-research` : default is *false*

`--omnitrace-build-from-source` : default is *false*

`--use-makefile` : default is 0

NOTE: distribution and distribution version are autodetected that is running the script.

It is possible to print these options to terminal by doing:

```
./bare_system/main_setup.sh --help
```

Bare System Installation Instructions – called by main_setup.sh 1/3

```
rocm/scripts/baseospackages_setup.sh
rocm/scripts/lmod_setup.sh
rocm/scripts/rocm_setup.sh --rocm-version ${ROCM_VERSION}
rocm/scripts/rocm_omnitrace_setup.sh --rocm-version ${ROCM_VERSION}
rocm/scripts/rocm_omniperf_setup.sh --rocm-version ${ROCM_VERSION}
-------------------------------------------------------------------------
comm/scripts/openmpi_setup.sh --rocm-version ${ROCM_VERSION} --amdgpu-gfxmodel ${AMDGPU_GFXMODEL}
comm/scripts/mpi4py_setup.sh --rocm-version ${ROCM_VERSION} --build-mpi4py ${BUILD_MPI4PY}
comm/scripts/mvapich2_setup.sh --rocm-version ${ROCM_VERSION}
-------------------------------------------------------------------------
tools/scripts/omnitrace_setup.sh --rocm-version ${ROCM_VERSION} --amdgpu-gfxmodel ${AMDGPU_GFXMODEL} \
    --omnitrace-build-from-source ${OMNITRACE_BUILD_FROM_SOURCE} --install-omnitrace-research
${INSTALL_OMNITRACE_RESEARCH}
tools/scripts/grafana_setup.sh
tools/scripts/omniperf_setup.sh --rocm-version ${ROCM_VERSION} --install-omniperf-research
${INSTALL_OMNIPERF_RESEARCH}
tools/scripts/hpctoolkit_setup.sh --rocm-version ${ROCM_VERSION} --build-hpctoolkit ${BUILD_HPCTOOLKIT}
tools/scripts/tau_setup.sh --rocm-version ${ROCM_VERSION} --build-tau ${BUILD_TAU}
tools/scripts/scorep_setup.sh --rocm-version ${ROCM_VERSION} --build-scorep ${BUILD_SCOREP}
```

Bare System Installation Instructions – called by main_setup.sh 2/3

```
extras/scripts/compiler_setup.sh

extras/scripts/apps_setup.sh

extras/scripts/apps_setup_basic.sh

extras/scripts/cupy_setup.sh --rocm-version ${ROCM_VERSION} --amdgpu-gfxmodel ${AMDGPU_GFXMODEL} \
    --build-cupy ${BUILD_CUPY}

extras/scripts/jax_setup.sh --rocm-version ${ROCM_VERSION} --amdgpu-gfxmodel ${AMDGPU_GFXMODEL} \
    --build-jax ${BUILD_JAX}

extras/scripts/pytorch_setup.sh --rocm-version ${ROCM_VERSION} --amdgpu-gfxmodel ${AMDGPU_GFXMODEL} \
    --build-pytorch ${BUILD_PYTORCH}

extras/scripts/kokkos_setup.sh --rocm-version ${ROCM_VERSION} --build-kokkos ${BUILD_KOKKOS}
```

Bare System Installation Instructions – called by main_setup.sh 3/3

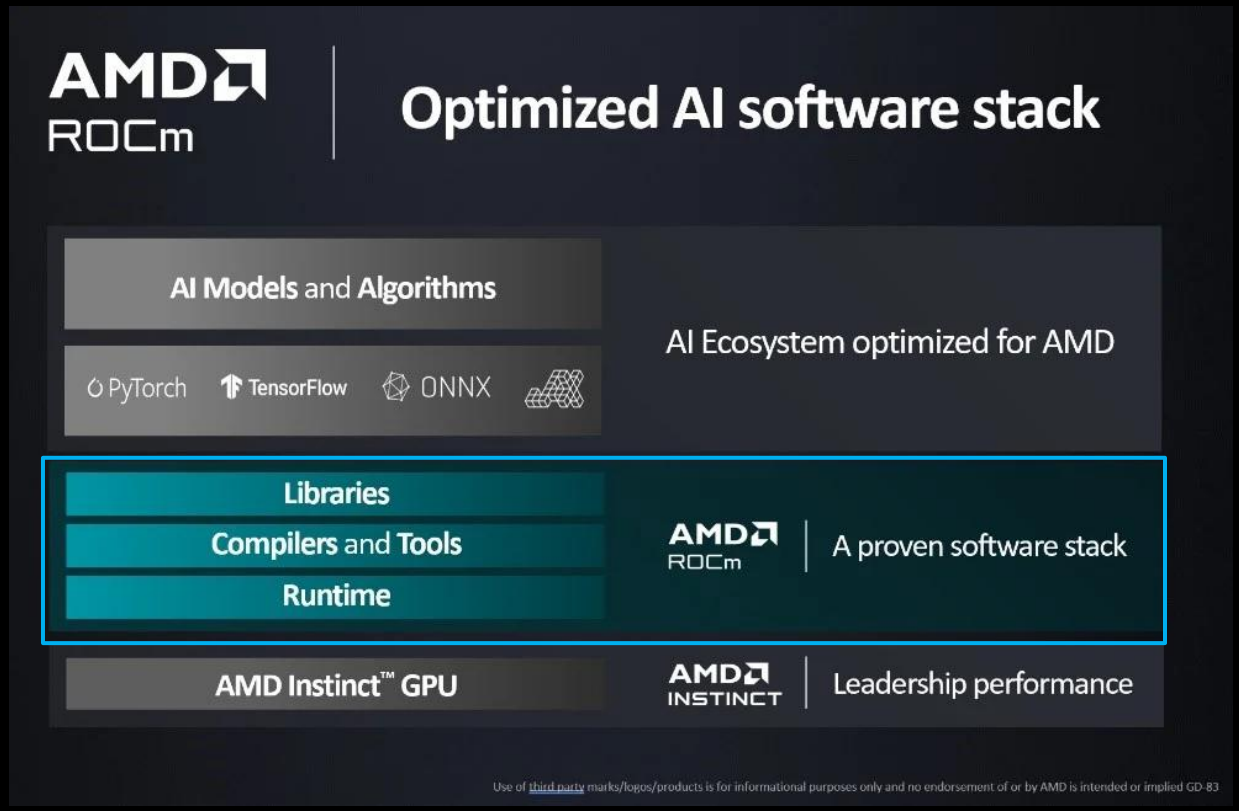
```
extras/scripts/miniconda3_setup.sh --rocm-version ${ROCM_VERSION} --build-miniconda3 ${BUILD_MINICONDA3} --python-version ${PYTHON_VERSION}

extras/scripts/miniforge3_setup.sh --rocm-version ${ROCM_VERSION} --build-miniforge3 ${BUILD_MINIFORGE3}

extras/scripts/hipfort_setup.sh --rocm-version ${ROCM_VERSION} --build-hipfort ${BUILD_HIPFORT}

extras/scripts/hipifly_setup.sh --rocm-version ${ROCM_VERSION} --hipifly-module ${HIPIFLY_MODULE} --hipifly-header-path extras/sources/hipifly/

extras/scripts/hdf5_setup.sh --rocm-version ${ROCM_VERSION} --build-hdf5 ${BUILD_HDF5}

extras/scripts/netcdf_setup.sh --rocm-version ${ROCM_VERSION} --build-netcdf ${BUILD_NETCDF}

extras/scripts/fftw_setup.sh --rocm-version ${ROCM_VERSION} --build-fftw ${BUILD_FFTW}

extras/scripts/x11vnc_setup.sh --build-x11vnc ${BUILD_X11VNC}
```

ROCm installation

Presenter: Bob Robey
Organization: AMD

ROCm Software stack

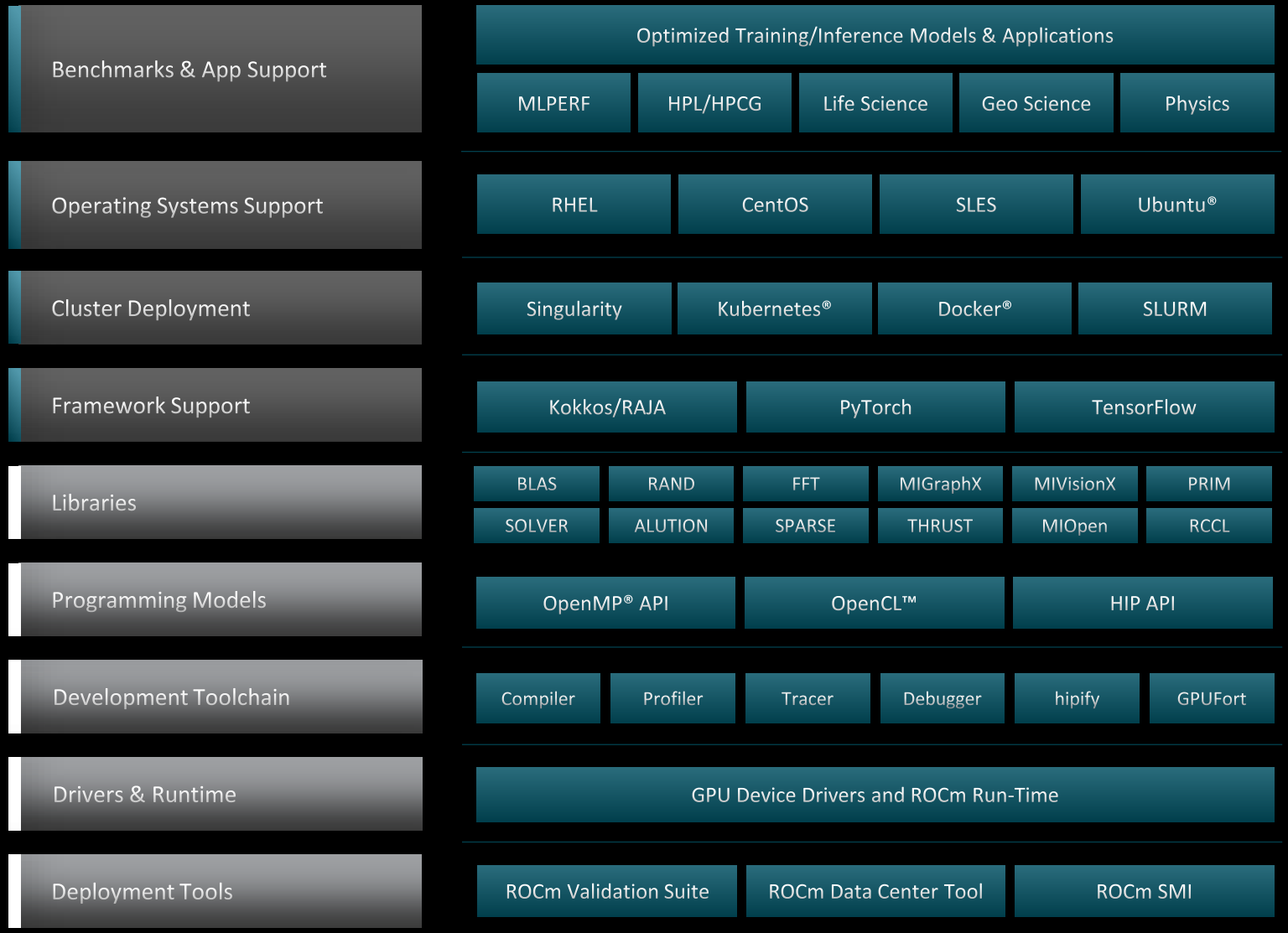
- ROCm is an **open-source** stack, composed primarily of open-source software:
  https://github.com/ROCm/ROCm
- Consists of a collection of runtime, compilers, development tools, and libraries that enable GPU programming from low level kernel to end user applications



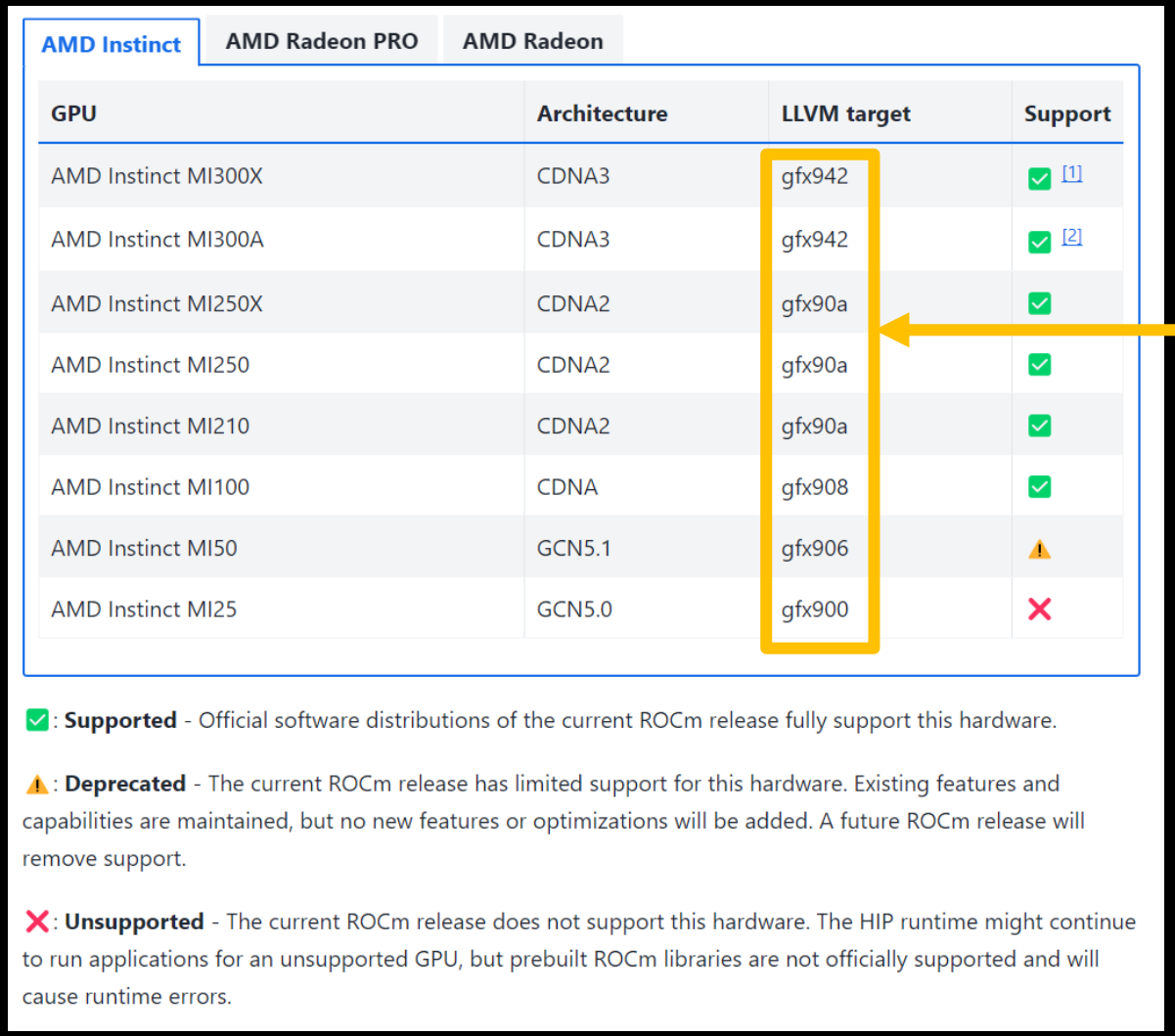
ROCm : Open Software Platform For GPU Compute

AMD ROCm

- Unlocked GPU Power To Accelerate Computational Tasks
- Optimized for HPC and Deep Learning Workloads at Scale
- Open Source Enabling Innovation, Differentiation, and Collaboration



ROCm Hardware Support (Linux®)



AMD Instinct | AMD Radeon PRO | AMD Radeon

GPU	Architecture	LLVM target	Support
AMD Instinct MI300X	CDNA3	gfx942	✅ [1]
AMD Instinct MI300A	CDNA3	gfx942	✅ [2]
AMD Instinct MI250X	CDNA2	gfx90a	✅
AMD Instinct MI250	CDNA2	gfx90a	✅
AMD Instinct MI210	CDNA2	gfx90a	✅
AMD Instinct MI100	CDNA	gfx908	✅
AMD Instinct MI50	GCN5.1	gfx906	⚠️
AMD Instinct MI25	GCN5.0	gfx900	❌

✅: **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.

NOTE: in the screenshot you can also see the **AMDGPU_GFXMODEL** for your architecture of interest

More details: https://rocm.docs.amd.com/projects/install-on-linux/en/latest/reference/system-requirements.html

ROCm System Support (Linux®)

Operating system	Kernel	Glibc	Support
Ubuntu 24.04.3	6.8 [GA], 6.14 [HWE]	2.39	✅
Ubuntu 22.04.5	5.15 [GA], 6.8 [HWE]	2.35	✅
RHEL 9.6	5.14.0-570	2.34	✅
RHEL 9.4	5.14.0-427	2.34	✅
RHEL 8.10	4.18.0-553	2.28	✅ [10]
SLES 15 SP7	6.4.0-150700.51	2.38	✅ [11]
Debian 12	6.1.0	2.36	✅ [11]
Rocky Linux 9	5.14.0-570	2.34	✅ [12]
Azure Linux 3.0	6.6.60	2.38	✅ [13]
Oracle Linux 9	5.15.0 (UEK)	2.34	✅ [14]
Oracle Linux 8	5.15.0 (UEK)	2.28	✅ [15]

To determine your operating system, type from terminal:

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

To determine your kernel, type from terminal:

```
uname -srmv
```

More details: https://rocm.docs.amd.com/projects/install-on-linux/en/latest/reference/system-requirements.html

Support for Operating Systems in installation scripts

- All supported **Ubuntu®** distributions. Releases 24.04 and 22.04 are tested

- **RHEL** compatible
  - Red Hat Enterprise Linux
    Rocky Linux
    Alma Linux

- **SUSE®** compatible
  - OpenSUSE/leap

ROCm Installation Options

There are four different options for installing the ROCm software stack

1. Single version installation of ROCm using an AMD provided install script
2. Installation of a ROCm version on a system where ROCm is already installed (support for multiple versions in one system)
3. Installation of ROCm using Ubuntu's apt-get
4. Package manager install

➢ We will only show the process for the AMD provided install script, using the script:

```
rocm/scripts/rocm_setup.sh
```

For information on the other methods see:

- AMD Lab-Notes article: https://gpuopen.com/learn/amd-lab-notes/amd-lab-notes-rocm-installation-readme/
- ROCm documentation: https://rocm.docs.amd.com/en/latest/deploy/linux/index.html

- This installation procedure installs
  - `hiplibsdk,rocmdev,rocmdevtools,lrt,openclsdk,openmpsdk,mlsdk`

  For a list of current software options, run:

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

ROCm Installation with AMD Script (details)

```
# Get the key for the ROCm software
wget -q -O - https://repo.radeon.com/rocm/rocm.gpg.key | gpg --dearmor | sudo tee
/etc/apt/keyrings/rocm.gpg > /dev/null

# Update package list
sudo DEBIAN_FRONTEND=noninteractive apt-get update


# Get the amdgpu-install script
wget -q https://repo.radeon.com/amdgpu-
install/${AMDGPU_ROCM_VERSION}/${DISTRO}/${ROCM_REPO_DIST}/amdgpu-
install_${AMDGPU_INSTALL_VERSION}_all.deb


# Run the amdgpu install script. We have already installed the kernel driver, so use --no-dkms. For a
typical installation, the HIP libraries and ROCm are recommended
sudo DEBIAN_FRONTEND=noninteractive apt-get install -q -y ./amdgpu-
install_${AMDGPU_INSTALL_VERSION}_all.deb
```

ROCm Installation with AMD Script (details)

```
# if ROCM_VERSION is greater than 6.1.2, the awk command will give the ROCM_VERSION number
# if ROCM_VERSION is less than or equal to 6.1.2, the awk command result will be blank
      result=`echo $ROCM_VERSION | awk '$1>6.1.2'` && echo $result

      if [[ "${result}" ]]; then # ROCM_VERSION >= 6.2
         result=`echo $DISTRO_VERSION | awk '$1>24.00'` && echo $result
         if [[ "${result}" ]]; then
            # rocm-asan not available in Ubuntu® 24.04
           amdgpu-install -q -y --usecase=hiplibsdk,rocmdev,rocmdevtools,lrt,openclsdk,openmpsdk,mlsdk --no-dkms
         else
            # removing asan to reduce image size
            #amdgpu-install -q -y --usecase=hiplibsdk,rocmdev,lrt,openclsdk,openmpsdk,mlsdk,asan --no-dkms
            amdgpu-install -q -y --usecase=hiplibsdk,rocmdev,,rocmdevtools,lrt,openclsdk,openmpsdk,mlsdk --no-dkms
         fi
      else # ROCM_VERSION < 6.2
         amdgpu-install -q -y --usecase=hiplibsdk,rocm --no-dkms
      fi
```

ROCm Post Installation Verification

Run **`rocm-smi`** to get information on the available GPUs (below showing a system with 8 MI250 GPUs)

```
========================================= ROCm System Management Interface =========================================
================================================= Concise Info =================================================
Device  [Model : Revision]    Temp    Power  Partitions      SCLK    MCLK     Fan  Perf  PwrCap  VRAM%  GPU%
        Name (20 chars)       (Edge)  (Avg)  (Mem, Compute)
============================================================================================================
0       [0x0b0c : 0x01]       42.0°C  83.0W  N/A, N/A        800Mhz  1600Mhz  0%   auto  560.0W    0%   0%
        AMD INSTINCT MI250 (
1       [0x0b0c : 0x01]       40.0°C  N/A    N/A, N/A        800Mhz  1600Mhz  0%   auto  0.0W      0%   0%
        AMD INSTINCT MI250 (
2       [0x0b0c : 0x01]       35.0°C  84.0W  N/A, N/A        800Mhz  1600Mhz  0%   auto  560.0W    0%   0%
        AMD INSTINCT MI250 (
3       [0x0b0c : 0x01]       38.0°C  N/A    N/A, N/A        800Mhz  1600Mhz  0%   auto  0.0W      0%   0%
        AMD INSTINCT MI250 (
4       [0x0b0c : 0x01]       36.0°C  88.0W  N/A, N/A        800Mhz  1600Mhz  0%   auto  560.0W    0%   0%
        AMD INSTINCT MI250 (
5       [0x0b0c : 0x01]       38.0°C  N/A    N/A, N/A        800Mhz  1600Mhz  0%   auto  0.0W      0%   0%
        AMD INSTINCT MI250 (
6       [0x0b0c : 0x01]       42.0°C  93.0W  N/A, N/A        800Mhz  1600Mhz  0%   auto  560.0W    0%   0%
        AMD INSTINCT MI250 (
7       [0x0b0c : 0x01]       37.0°C  N/A    N/A, N/A        800Mhz  1600Mhz  0%   auto  0.0W      0%   0%
        AMD INSTINCT MI250 (
============================================================================================================
========================================= End of ROCm SMI Log ==============================================
```

ROCm Post Installation Verification

Run `amd-smi +options` to get further information

Output of `amd-smi monitor`:

GPU	POWER	GPU_TEMP	MEM_TEMP	GFX_UTIL	GFX_CLOCK	MEM_UTIL	MEM_CLOCK	ENC_UTIL	ENC_CLOCK	DEC_UTIL	DEC_CLOCK	THROTTLE	SINGLE_ECC	DOUBLE_ECC	PCIE_
REPLAY	VRAM_USED	VRAM_TOTAL	PCIE_TX	PCIE_RX											
0	83 W	42 °C	50 °C	0 %	800 MHz	0 %	1600 MHz	N/A	1000 MHz	N/A	875 MHz	UNTHROTTLED	0	0	
0	10 MB	65501 MB	0 Mb/s	0 Mb/s											
1	0 W	44 °C	51 °C	0 %	800 MHz	0 %	1600 MHz	N/A	1000 MHz	N/A	875 MHz	UNTHROTTLED	0	0	
0	10 MB	65501 MB	0 Mb/s	0 Mb/s											
2	84 W	37 °C	44 °C	0 %	800 MHz	0 %	1600 MHz	N/A	1000 MHz	N/A	875 MHz	UNTHROTTLED	0	0	
0	10 MB	65501 MB	0 Mb/s	0 Mb/s											
3	0 W	38 °C	44 °C	0 %	800 MHz	0 %	1600 MHz	N/A	1000 MHz	N/A	875 MHz	UNTHROTTLED	0	0	
0	10 MB	65501 MB	0 Mb/s	0 Mb/s											
4	88 W	39 °C	46 °C	0 %	800 MHz	0 %	1600 MHz	N/A	1000 MHz	N/A	875 MHz	UNTHROTTLED	0	0	
0	10 MB	65501 MB	0 Mb/s	0 Mb/s											
5	0 W	40 °C	46 °C	0 %	800 MHz	0 %	1600 MHz	N/A	1000 MHz	N/A	875 MHz	UNTHROTTLED	0	0	
0	10 MB	65501 MB	0 Mb/s	0 Mb/s											
6	92 W	43 °C	51 °C	0 %	800 MHz	0 %	1600 MHz	N/A	1000 MHz	N/A	875 MHz	UNTHROTTLED	0	0	
0	10 MB	65501 MB	0 Mb/s	0 Mb/s											
7	0 W	40 °C	52 °C	0 %	800 MHz	0 %	1600 MHz	N/A	1000 MHz	N/A	875 MHz	UNTHROTTLED	0	0	
0	10 MB	65501 MB	0 Mb/s	0 Mb/s											

Output of `amd-smi version`:

```
AMDSMI Tool: 24.4.1+afcd367 | AMDSMI Library version: 24.5.1.0 | ROCm version: 6.1.0
```

Modules

Presenter: Bob Robey
Organization: AMD

Module Installation – lua, lmod, modules

To set up modules, first need to install `lmod` and then setup the module files. These are set at the end of each script to install a given library. We are going to use the script:

`rocm/scripts/lmod_setup.sh`

Install lua – using the package manager:

```
sudo DEBIAN_FRONTEND=noninteractive apt-get -qqy install lmod
```

Add directories to module path – these are in the `/etc/lmod/modulespath` file:

```
sudo sed -i -e
'1,$s!/etc/lmod/modules!/etc/lmod/modules/Linux\n/etc/lmod/modules/ROCm\n/etc/lmod/modules/ROCmPlus\n/etc/lmod/modules/ROCmPlus-
MPI\n/etc/lmod/modules/ROCmPlus-AMDResearchTools\n/etc/lmod/modules/ROCmPlus-LatestCompilers\n/etc/lmod/modules/ROCmPlus-
AI\n/etc/lmod/modules/misc!' /etc/lmod/modulespath
cat /etc/lmod/modulespath
```

Set up default environment:

```
cat << EOF | sudo tee -a /etc/bash.bashrc
   if ! shopt -q login_shell; then
     if [ -d /etc/profile.d ]; then
       for i in /etc/profile.d/*.sh; do
         if [ -r \$i ]; then
           . \$i
         fi
       done
     fi
   fi
EOF
sudo ln -s /usr/share/lmod/6.6/init/profile /etc/profile.d/z00_lmod.sh
sudo ln -s /usr/share/lmod/6.6/init/cshrc /etc/profile.d/z00_lmod.csh
```

Inspect the Modules Available

It is possible to inspect the modules available in the system by doing:

```
module avail
```

```
---------------------------------------------------------- /etc/lmod/modules/Linux ----------------------------------------------------------
   clang/base    gcc/base

---------------------------------------------------------- /etc/lmod/modules/LinuxPlus ----------------------------------------------------------
   miniconda3/25.3.1    miniforge3/24.9.0

------------------------------------------------------------ /etc/lmod/modules/ROCm -------------------------------------------------------------
   amdclang/19.0.0-6.4.3          hipfort/6.4.3    rocm/6.4.3                      rocprofiler-sdk/6.4.3
   amdflang-new/rocm-afar-7.0.5    opencl/6.4.3     rocprofiler-compute/6.4.3 (D)    rocprofiler-systems/6.4.3 (D)

---------------------------------------------------------- /etc/lmod/modules/ROCmPlus ----------------------------------------------------------
   adios2/2.10.1    hdf5/1.14.6                  hypre/2.33.0     netcdf-c/4.9.3          petsc/3.23.1    tau/dev
   fftw/3.3.10      hpctoolkit/2024.01.99-next    kokkos/4.6.01    netcdf-fortran/4.6.2    scorep/9.0

------------------------------------------------------- /etc/lmod/modules/ROCmPlus-MPI -------------------------------------------------------
   mpi4py/4.0.3    openmpi/5.0.7-ucc1.4.4-ucx1.18.1

---------------------------------------------------- /etc/lmod/modules/ROCmPlus-AMDResearchTools -----------------------------------------------------
   rocprofiler-compute/develop    rocprofiler-systems/develop

---------------------------------------------------- /etc/lmod/modules/ROCmPlus-LatestCompilers -----------------------------------------------------
   hipfort_from_source/6.4.3

------------------------------------------------------- /etc/lmod/modules/ROCmPlus-AI --------------------------------------------------------
   cupy/13.6.0    hip-python/13.6.0    pytorch/2.8.0_tunableop_enabled        tensorflow/merge-250318
   ftorch/dev     jax/0.6.0            pytorch/2.8.0                   (D)

---------------------------------------------------------- /etc/lmod/modules/misc ----------------------------------------------------------
   hipifly/dev

  Where:
   D:  Default Module
```

About Modules

- Linux®
  - The compilers listed here are alternates to the default compiler for the Ubuntu® distribution

- ROCm – break up modules for ROCm install into four modules
  - rocm – adds path to ROCm
  - amdclang – adds path to LLVM™ and sets compiler CC, CXX, FC
  - hipfort – adds path to hipfort
  - opencl – adds path to OpenCL™
  - omnitrace and omniperf – only available for ROCm version 6.2

- ROCmPlus

  These modules are dependent on the ROCm version, but need to be installed individually after the ROCm install
  - ROCmPlus-AI
    - ML/AI frameworks available at the moment: hip-python, CuPy, Jax, PyTorch, and Ftorch
  - ROCmPlus-MPI
    - three GPU-aware MPIs are available: OpenMPI, Mvapich2, and MPI4Py
  - ROCmPlus-AMDResearchTools
    - two profiling tools out of AMD Research are available: Rocprof-systems/develop and Rocprof-compute/develop

Other Relevant Module Commands

- `module list`
  shows currently loaded modules

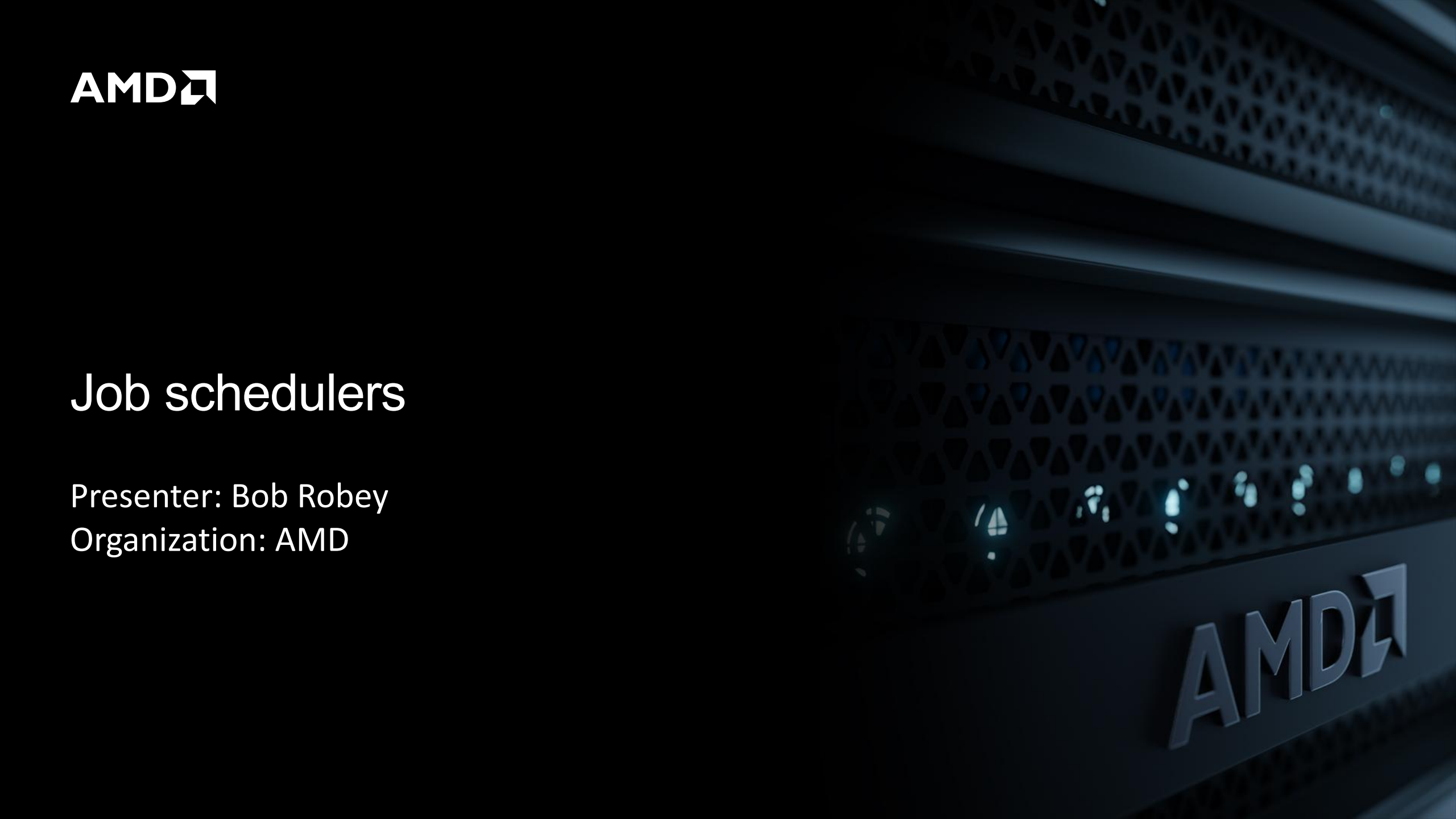
- `module load <module>`
  loads a specific module

- `module unload <module>`
  unloads a specific module

- `module purge`
  unloads all currently loaded modules

- `module show <module>`
  shows the content of a module file (example: `module show cupy`):

```
whatis("HIP version of CuPy")
load("rocm/6.1.0")
prepend_path("PYTHONPATH","/opt/rocmplus-6.1.0/cupy")
```



Job schedulers

Presenter: Bob Robey
Organization: AMD

Slurm Installation

- The first step for a slurm installation is to create a slurm configuration file. Fortunately, there is a tool to help with this process at https://slurm.schedmd.com/configurator.easy.html. Some experimentation is necessary to determine how to fill in some of the fields.
- The Training Container queries the system and tries to set up the proper configuration in the init.sh script run when the container starts.
- Install Slurm dependencies

  sudo apt-get install -y libpmi2-0-dev
- Install Slurm

  sudo apt-get install -y slurmd slurmctld
- Add slurm.conf file to /etc/slurm directory

```
sudo mkdir /etc/slurm && chmod 777 /etc/slurm
sudo cp slurm.conf /etc/slurm
sudo chown slurm:slurm /etc/slurm/slurm.conf
```

- Add munge options

```
echo "OPTIONS=\"--force --key-file /etc/munge/munge.key --num-threads 10\"" > /etc/default/munge
```

Testing the success of the installation

Presenter: Bob Robey
Organization: AMD

Testing the Installation

A test suite to test the installation of the software is available at:

https://github.com/amd/HPCTrainingExamples.git

There are currently two ways to test the success of the installation:

1. Directly: clone the repo and run the test suite
   - This option can be used both with the training container and with the bare system install scripts, with either the `main_setup.sh` (when performing the actual installation) or with the `test_install.sh` (when testing the installation before deployment)

2. With the Makefile build of the test installation: run `make <package>` followed by `make <package_tests>`
   - Note that this option only applies when doing a test installation using `test_install.sh` and specifying the `--use-makefile` input flag when launching the script

Testing the Installation Directly

To test the installation directly, do:

```
git clone https://github.com/amd/HPCTrainingExamples.git
cd HPCTrainingExample/tests
./runTests.sh --test
```

If no `--test` is specified, **all tests** will be run.

To run OpenMPI tests do `./runTests.sh --openmpi`

Specific to OpenMPI and Mvapich2 we have two sets of tests:

1. **Hello World Tests**
   - To test basic MPI functionalities (`MPI_Init`, `MPI_Comm_size`, `MPI_Comm_rank`, `MPI_Finalize`, etc.) and make sure that the compilers loaded by the library are used instead of those from the system

2. **OSU Mini Benchmarks:** https://mvapich.cse.ohio-state.edu/benchmarks/

Testing the Installation with Makefile

To test the installation using the Makefile run:

```
git clone https://github.com/AMD/HPCTrainingDock
        cd HPCTrainingDock
                ./bare_system/test_install.sh --distro ubuntu --distro-versions 22.04
                                              --rocm-version 6.3.2 --use-makefile 1
```

NOTE: if `--distro` and `--distro-versions` are left out, the test install script will detect the current distro and distro version **on the system where the script is being run** and use that. If `--rocm-version` is left out, the script also tries to detect the current ROCm version on your system and **use that as default**

As explained, the above script will automatically get you into a container as `sysdamin`
Once in the container do:

```
make <package>
make <package_tests>
```

For instance, for CuPy:

```
make cupy
make cupy_tests
```

Create a Pre-built Binary Distribution of ROCm

It is possible to create a binary distribution of ROCm 6.4.1 by taring up the `rocm-6.4.1` directory
Then, the next build will restore from the tar file
This can reduce the build time for the subsequent test installs. To test this, do:

```
git clone https://github.com/AMD/HPCTrainingDock
cd HPCTrainingDock
bare_system/test_install.sh --distro ubuntu --distro-versions 24.04 --rocm-version
6.4.1 --use-makefile 1
make rocm_package
```

This make command tars up the `rocm-6.4.1` directory and then the next build it will restore from the tar file

NOTE: MPCDF uses this approach to create the desired ROCm installation tarball, then copy it from the Docker® container to their installation framework.

Final remarks

Presenter: Bob Robey
Organization: AMD

Other Resources

- Spack build recipes
  - We used these with the HPCViewer and PDT install

- E4S – Continuous testing system at University of Oregon for the Exascale Computing Project
- Infinity Hub: https://github.com/amd/InfinityHub-CI
- ROCm docs at https://rocm.docs.amd.com
- ROCm blog posts
  - Installing various AI/ML packages
- Lab Notes posted at https://gpuopen.com (an AMD website)
  - Installing ROCm
  - Installing GPU-Aware MPIs

Conclusion

- Support for standardized software installations can help with portability
- Optimized software installations can improve performance over generic installations
- Quick, streamlined software installation can help get systems up quickly
- Alternate system administration workflows such as different installation directories and reducing root users
- Application developers can test different ROCm and Operating System versions
- Application developers can set up local workstations for Continuous Integration and local development

Disclaimer

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions, and typographical errors. The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. Any computer system has risks of security vulnerabilities that cannot be completely prevented or mitigated. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

THIS INFORMATION IS PROVIDED ‘AS IS.” AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS, OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION. AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY RELIANCE, DIRECT, INDIRECT, SPECIAL, OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Third-party content is licensed to you directly by the third party that owns the content and is not licensed to you by AMD. ALL LINKED THIRD-PARTY CONTENT IS PROVIDED “AS IS” WITHOUT A WARRANTY OF ANY KIND. USE OF SUCH THIRD-PARTY CONTENT IS DONE AT YOUR SOLE DISCRETION AND UNDER NO CIRCUMSTANCES WILL AMD BE LIABLE TO YOU FOR ANY THIRD-PARTY CONTENT. YOU ASSUME ALL RISK AND ARE SOLELY RESPONSIBLE FOR ANY DAMAGES THAT MAY ARISE FROM YOUR USE OF THIRD-PARTY CONTENT.


AMD