Matthew Sgambati
HPC System Administrator

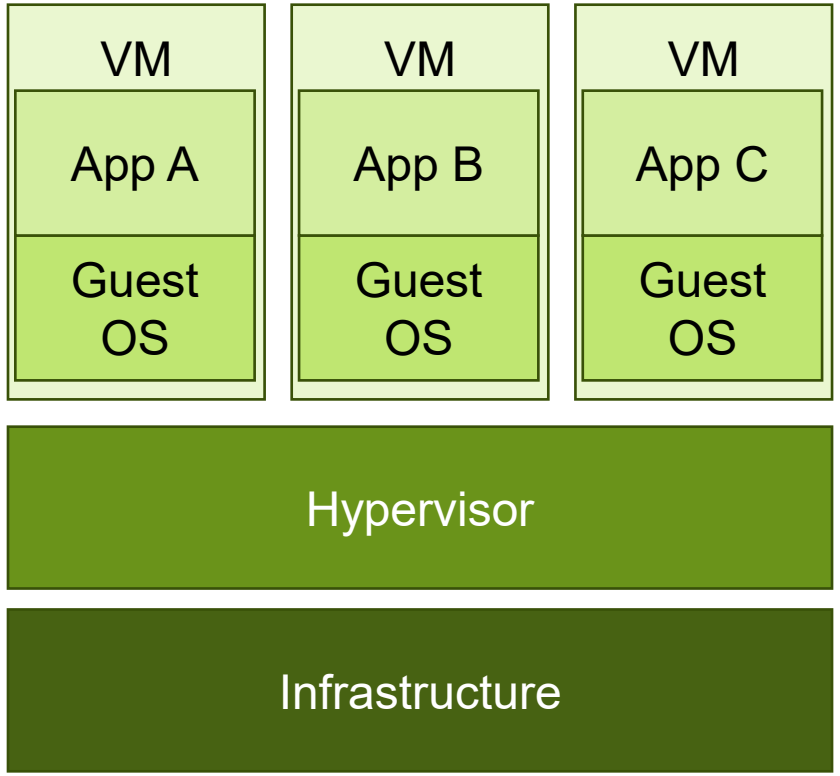
Containers

INL/MIS-23-75868

Idaho National Laboratory

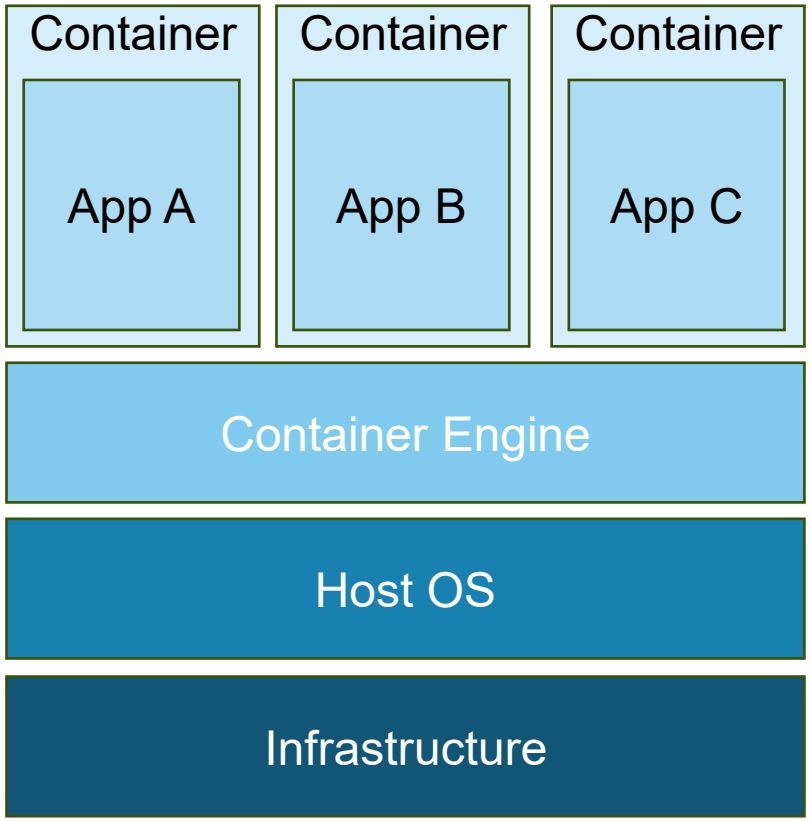
What a container is not?

- Not a Virtual Machine
- Virtual Machines
  - Heavy Weight
  - Full copy of OS
  - Physical hardware abstraction
    - Hardware-level virtualization



What a container is?

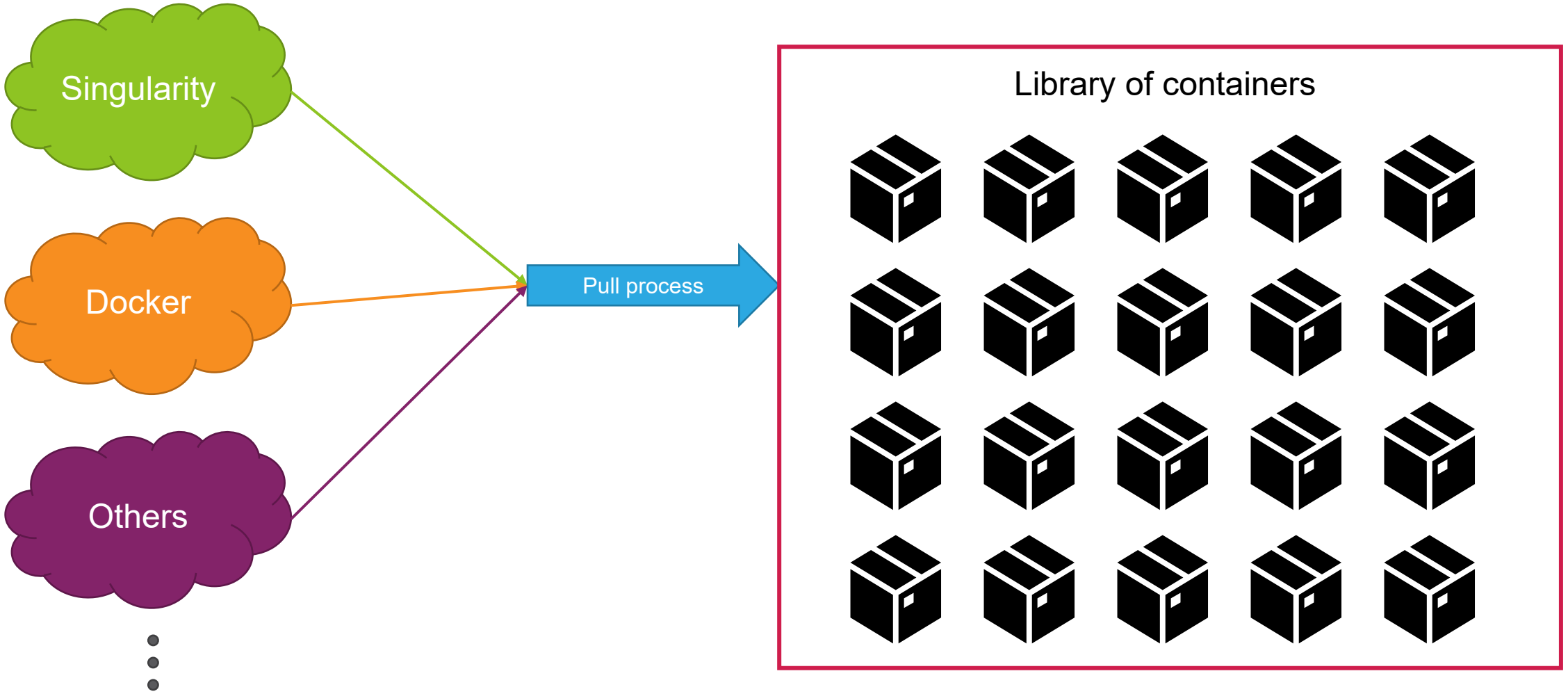
- User Defined Software Stack
  - All components necessary for the application are included in the container, including the OS
- Containers
  - Light Weight
  - Share components of Host OS
  - Application layer abstraction
    - OS virtualization



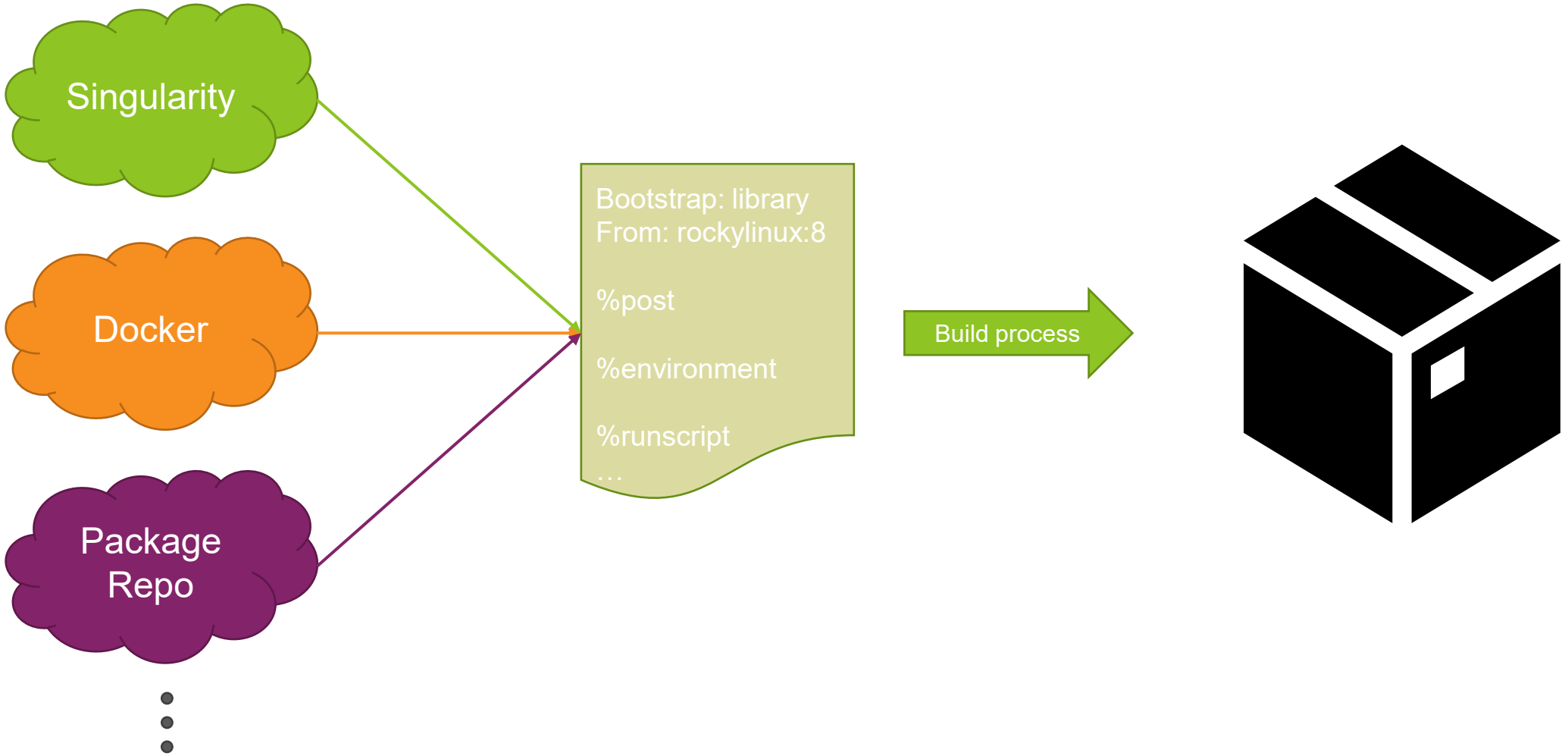
Why containers?

- Verification and Validation: Is my code running correctly still? Is it passing all unit tests and giving the same results as my last few papers?
- Reproducibility: Can I run my same application 10 years later? Can I rebuild my container 10 years later?
- Portability across multiple supercomputer architectures: Can I run the same container on multiple supercomputers with different networks and drivers *without rebuilding and without sacrificing performance*?
- Simplification: one environment, customized packages (independence from system administrators); make it easier on the user: Can I work without needing intervention from the system administrator?
- Security (insulation against package versioning issues): Does a security update on a supercomputer break my code?

Containers prebuilt



Containers with def file

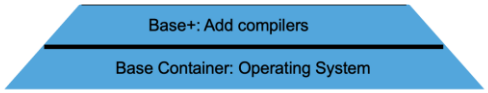


Base Container Definition File

Base Container: Operating System

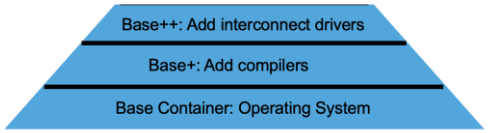
```
Bootstrap: docker
From: rockylinux:8.6

%post
    # Capture useful system information
    echo "Architecture" $(lscpu | grep "^Architecture" | cut -d ':' -f 2) >> "${SINGULARITY_LABELS}"
    echo "CPU" $(lscpu | grep "^Model name" | cut -d ':' -f 2) >> "${SINGULARITY_LABELS}"
    echo "uname" $(uname -srvpio) >> "${SINGULARITY_LABELS}"

%test
    if grep -q 'NAME="Rocky Linux"' /etc/os-release; then
        echo "SUCCESS: Container base is Rocky Linux as expected."
    else
        echo "ERROR: Container base is not Rocky Linux."
        exit 1
    fi

%labels
    Authors Matthew.Sgambati@inl.gov Matthew.Anderson2@inl.gov
    Version 1.0.0

%help
    Rocky Linux 8.6 Base Container

```

Base+ Container Definition File



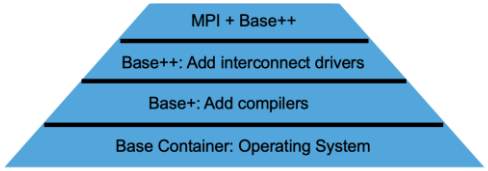

```
Bootstrap: oras
From: <container_registry>/hpcbase/base_00:1.0.0

%post
    # Change repos to point to local static mirror
    sed -i 's/^mirrorlist/#mirrorlist/' /etc/yum.repos.d/Rocky-*.repo
    sed -i 's#.*baseurl=http://dl.rockylinux.org/\$contentdir#baseurl=http://<local_static_mirror>/repos/rocky-linux/20221208#'
    ↪  /etc/yum.repos.d/Rocky-*.repo

    dnf clean all
    dnf makecache

    # Install commonly used packages for building code and modifiying files
    dnf install -y bzip2 gcc gcc-gfortran gcc-c++ gdb git make python39 python39-pip python39-setuptools tar vim
    dnf clean all

    # Capture useful system information
    echo "Architecture" $(lscpu | grep "^Architecture" | cut -d ':' -f 2) >> "${SINGULARITY_LABELS}"
    echo "CPU" $(lscpu | grep "^Model name" | cut -d ':' -f 2) >> "${SINGULARITY_LABELS}"
    echo "uname" $(uname -srvpio) >> "${SINGULARITY_LABELS}"

%test
    GCC=$(which gcc)
    if [ $? -eq 0 ]; then
        echo "SUCCESS: gcc is available at ${GCC}"
    else
        echo "ERROR: gcc is not installed"
        exit 1
    fi

%labels
    Authors Matthew.Sgambati@inl.gov Matthew.Anderson2@inl.gov
    Version 1.0.0

%help
    Rocky Linux 8.6 Base Container
    Extra dependencies for building code and modifiying files are installed in this layer.

```

Base++ Container Definition File




```
Bootstrap: oras
From: <container_registry>/hpcbase/extra_01:1.0.0

%post
    # Create environment variables of software versions
    MLNX_OFED=MLNX_OFED_LINUX-5.4-3.6.8.1-rhel8.6-x86_64
    OPX=CornelisOPX-Basic.RHEL86-x86_64.10.12.1.0.7

    # Make src in /opt to hold source code
    mkdir -p /opt/src

    # Download the MLNX_OFED and OPX files to /opt
    cd /opt
    curl -O http://<local_static_mirror>/interconnects/MLNX/20221208/${MLNX_OFED}.tgz
    curl -O http://<local_static_mirror>/interconnects/OPX/20230212/${OPX}.tgz

    # Install required packages for InfiniBand
    dnf install -y python39 pciutils lsof ethtool tcsh libnl3 tk numactl-libs tcl

    # Extract MLNX_OFED and install it
    tar xf ${MLNX_OFED}.tgz -C /opt/src
    /opt/src/${MLNX_OFED}/mlnxofedinstall --without-fw-update --skip-unsupported-devices-check --basic --user-space-only --distro RHEL8.6
    ↪  --without-depcheck -q

    # Install required packages for OmniPath
    dnf install -y irqbalance kernel-modules-extra kmod libgcc perl perl-Getopt-Long perl-Socket opensm-libs python2 libatomic
    dnf download ibacm*x86_64
    rpm -ivh --nodeps ibacm*.rpm

    # Extract OPX and install it
    tar xf ${OPX}.tgz -C /opt/src
    cd /opt/src/${OPX}
    ./INSTALL -i intel_hfi -i opa_stack --user-space

    # Clean up tarballs/downloads and source directories
    cd /opt
    rm -rf /opt/src
    rm -f /opt/*.tgz
    rm -f /opt/*.rpm

    # Capture useful system information
    echo "Architecture" $(lscpu | grep "^Architecture" | cut -d ':' -f 2) >> "${SINGULARITY_LABELS}"
    echo "CPU" $(lscpu | grep "^Model name" | cut -d ':' -f 2) >> "${SINGULARITY_LABELS}"
    echo "uname" $(uname -srvpio) >> "${SINGULARITY_LABELS}"

```

MPI + Base++ Container Definition File




```
Bootstrap: oras
From: <container_registry>/hpcbase/interconnect_02:1.0.0

%post
    # Install required packages
    dnf install -y hwloc findutils

    # Create directories to store files and source code
    mkdir -p /opt/mpi/examples
    mkdir -p /opt/src
    mkdir -p /opt/tars

    # Create mpitest.c program from here doc
    cat <<- EOF > /opt/mpi/examples/mpitest.c
        #include <mpi.h>
        #include <stdio.h>
        #include <stdlib.h>

        int main (int argc, char **argv) {
                int rc;
                int size;
                int myrank;

                rc = MPI_Init (&argc, &argv);
                if (rc != MPI_SUCCESS) {
                    fprintf (stderr, "MPI_Init() failed");
                    return EXIT_FAILURE;
                }

                rc = MPI_Comm_size (MPI_COMM_WORLD, &size);
                if (rc != MPI_SUCCESS) {
                    fprintf (stderr, "MPI_Comm_size() failed");
                    goto exit_with_error;
                }

                rc = MPI_Comm_rank (MPI_COMM_WORLD, &myrank);
                if (rc != MPI_SUCCESS) {
                    fprintf (stderr, "MPI_Comm_rank() failed");
                    goto exit_with_error;
                }

                fprintf (stdout, "Hello, I am rank %d/%d\n", myrank, size);

                MPI_Finalize();

                return EXIT_SUCCESS;

    # Install MPICH
    MPICH_VERSION=3.4.3
    MPICH_NAME="mpich-${MPICH_VERSION}"
    MPICH_URL="http://<local_static_mirror>/mpi/mpich/20221208/${MPICH_NAME}.tar.gz"
    MPICH_DIR="/opt/mpi/${MPICH_NAME}"

    echo "Installing MPICH-${MPICH_VERSION}..."
    ## Download
    cd /opt/tars
    curl -O ${MPICH_URL}
    tar xf ${MPICH_NAME}.tar.gz -C /opt/src

    ## Compile and install
    cd /opt/src/${MPICH_NAME}
    mkdir build
    cd build
    ../configure --prefix=${MPICH_DIR} --with-ucx=${UCX_DIR}
    make -j 16 |& tee log.make
    make check |& tee log.make_check
    make install |& tee log.make_install

    # Install OpenMPI
    OPENMPI_VERSION=4.1.4
    OPENMPI_NAME="openmpi-${OPENMPI_VERSION}"
    OPENMPI_DIR="/opt/mpi/${OPENMPI_NAME}"
    OPENMPI_URL="http://<local_static_mirror>/mpi/openmpi/20221208/${OPENMPI_NAME}.tar.gz"

    echo "Installing OPENMPI-${OPENMPI_VERSION}..."
    ## Download
    cd /opt/tars
    curl -O ${OPENMPI_URL}
    tar xf ${OPENMPI_NAME}.tar.gz -C /opt/src

    ## Compile and install
    cd /opt/src/${OPENMPI_NAME}
    mkdir build
    cd build
    ../configure --prefix=${OPENMPI_DIR} --with-ucx=${UCX_DIR}
    make -j 16 |& tee log.make
    make check |& tee log.make_check
    make install |& tee log.make_install
```

Questions?