Apptainer User Guide for R on VoWa HPC

This guide explains how to use Apptainer (formerly Singularity) containers to run R scripts and build custom R environments on the VoWa HPC cluster.

What is Apptainer?

Apptainer (formerly Singularity) is a container platform designed for HPC environments. It allows you to:

- Run software in isolated, reproducible environments
- Use pre-built images with all dependencies included
- Build custom images with your specific f. e. R packages
- Share computational environments with collaborators
- Ensure reproducibility across different systems

Why Use Containers?

- **Reproducibility**: Same environment every time
- **Isolation**: No conflicts with system packages
- **Portability**: Works on any system with Apptainer
- **Dependency Management**: All packages bundled together
- **Version Control**: Pin specific software versions

Quick Start: Running R Scripts

Running R Scripts in a Container

The basic command to run an R script using the RStudio container:

```
apptainer exec /data/apps/rstudio.sif Rscript your_script.R
```

Example: Run a simple R script

```
# Create a test script
cat > hello.R << 'EOF'
print("Hello from Apptainer!")
print(paste("R version:", R.version.string))
print(paste("Working directory:", getwd()))
EOF

# Run it with Apptainer
apptainer exec /data/apps/rstudio.sif Rscript hello.R
```

Running R Scripts with Arguments

Pass arguments to your R script:

```
apptainer exec /data/apps/rstudio.sif Rscript analysis.R arg1 arg2 arg3
```

In your R script (`analysis.R`):

```
# Read command line arguments
args <- commandArgs(trailingOnly = TRUE)
print(paste("Argument 1:", args[1]))
print(paste("Argument 2:", args[2]))
```

Running in a SLURM Job

Create a job script (`job.sh`):

```
#!/bin/bash
#SBATCH --job-name=r_analysis
#SBATCH --time=01:00:00
#SBATCH -c 4
#SBATCH --mem=8G

# Navigate to your working directory
cd $SLURM_SUBMIT_DIR

# Run R script in container
```

```
apptainer exec /data/apps/rstudio.sif Rscript my_analysis.R
```

Submit the job:

```
sbatch job.sh
```

Using Apptainer Interactively

Start an Interactive R Session

Launch R inside the container:

```
apptainer exec /data/apps/rstudio.sif R
```

You'll get an R prompt where you can work interactively:

```
R version 4.x.x ...
> library(tidyverse)
> data <- read.csv("mydata.csv")
> summary(data)
> quit()
```

Interactive Shell Inside Container

Start a shell inside the container to explore:

```
apptainer shell /data/apps/rstudio.sif
```

Now you're inside the container and can run any command:

```
Apptainer> which R
Apptainer> R --version
Apptainer> ls /usr/local/lib/R/site-library/
Apptainer> exit
```

Interactive Session on Compute Node

For longer interactive work, request a compute node first:

```
# Request interactive node
srun --pty --mem=16G --cpus-per-task=4 --time=02:00:00 bash

# Once on the compute node, start R
apptainer exec /data/apps/rstudio.sif R
```

Available Container Images

System-Wide Images

Located in `/data/apps/`:

RStudio Container

```
/data/apps/rstudio.sif
```

Includes:

- Latest R version
- RStudio Server
- tidyverse packages
- Common R packages
- Development tools

Usage:

```
apptainer exec /data/apps/rstudio.sif Rscript script.R
apptainer exec /data/apps/rstudio.sif R
```

Your Custom Images

Located in `$HOME/apps/`:

After building custom images (see Building Custom Images), they're saved to:

```
$HOME/apps/rstudio_custom.sif
$HOME/apps/my_analysis.sif
```

Usage:

```
apptainer exec $HOME/apps/rstudio_custom.sif Rscript script.R
```

Checking What's in a Container

List installed R packages:

```
apptainer exec /data/apps/rstudio.sif R -e 'installed.packages()[,c("Package","Version")]'
```

Check R version:

```
apptainer exec /data/apps/rstudio.sif R --version
```

Check installed system packages:

```
apptainer exec /data/apps/rstudio.sif dpkg -l | grep -i gdal
```

Building Custom Images

Why Build Custom Images?

- Install additional R packages not in the base image
- Add system dependencies (GDAL, PROJ, JAGS, etc.)
- Create reproducible environments for specific projects
- Share your environment with collaborators

Using the Job Template

The easiest way to build custom images is using the OpenOnDemand job template:

1. **Access Job Composer**:
   - Log in to OpenOnDemand
   - Go to **Jobs** → **Job Composer**

2. **Create job from template**:
   - Click **New Job → From Template**
   - Select "**Build Custom Apptainer Image**"
3. **Edit the definition file** (`rstudio_custom.def`):

```
Bootstrap: docker
From: rocker/rstudio:latest

%post
    # Install system dependencies
    apt-get update
    apt-get install -y libgdal-dev libproj-dev

    # Install R packages
    R --slave -e 'install.packages(c(
        "sf",
        "terra",
        "raster",
        "your-package-here"
    ), repos="https://cloud.r-project.org/")'
```

4. **Submit the job**:
   - Click **Submit**
   - Wait 1-4 hours for build to complete
5. **Use your custom image**:

```
apptainer exec $HOME/apps/rstudio_custom.sif Rscript script.R
```

Definition File Examples

Example 1: Spatial Analysis Environment

```
Bootstrap: docker
From: rocker/rstudio:latest

%post
    # Spatial dependencies
    apt-get update
    apt-get install -y \
        libgdal-dev \
```

```
        libproj-dev \
        libudunits2-dev \
        libgeos-dev

    # Spatial R packages
    R --slave -e 'install.packages(c(
        "sf",
        "terra",
        "raster",
        "rgdal",
        "sp",
        "lwgeom",
        "stars"
    ), repos="https://cloud.r-project.org/")'
```

Example 2: Bayesian Statistics Environment

```
Bootstrap: docker
From: rocker/rstudio:latest

%post
    # Install JAGS for Bayesian modeling
    apt-get update
    apt-get install -y jags

    # Bayesian R packages
    R --slave -e 'install.packages(c(
        "rjags",
        "jagsUI",
        "R2jags",
        "coda",
        "MCMCvis",
        "nimble",
        "brms",
        "rstanarm"
    ), repos="https://cloud.r-project.org/")'
```

Example 3: Bioinformatics Environment

```
Bootstrap: docker
From: rocker/rstudio:latest

%post
    apt-get update
    apt-get install -y \
        libcurl4-openssl-dev \
        libssl-dev \
        libxml2-dev

    # Install Bioconductor
    R --slave -e '
        if (!requireNamespace("BiocManager", quietly = TRUE))
            install.packages("BiocManager")
        BiocManager::install(c(
            "GenomicRanges",
            "DESeq2",
            "edgeR",
            "limma",
            "GenomicFeatures"
        ))
    '
```

Building Multiple Images

You can build different images for different projects:

1. Create separate definition files:
   - `spatial_analysis.def`
   - `bayesian_stats.def`
   - `bioinformatics.def`
2. Modify `script.sh` to set output name:

   ```
   OUTPUT_IMAGE="${HOME}/apps/spatial_analysis.sif"
   ```

3. Submit separate build jobs for each image

Best Practices

File Access in Containers

Apptainer automatically mounts your home directory and current working directory.

Accessible locations:

- `$HOME` - Your home directory
- Current working directory
- `/data` - Data storage (if available)
- `/scratch` - Temporary storage (if available)

Example:

```
# These work automatically
cd /home/username/project
apptainer exec /data/apps/rstudio.sif Rscript analysis.R

# Reads/writes files in /home/username/project
```

Binding Additional Directories

If you need access to other directories:

```
apptainer exec --bind /scratch:/scratch /data/apps/rstudio.sif Rscript script.R
```

Or set environment variable:

```
export APPTAINER_BIND="/scratch:/scratch,/data:/data"
apptainer exec /data/apps/rstudio.sif Rscript script.R
```

Working Directory

Always change to your data directory before running:

```
cd /path/to/your/data
apptainer exec /data/apps/rstudio.sif Rscript analysis.R
```

Or in a SLURM job:

```
cd $SLURM_SUBMIT_DIR
apptainer exec /data/apps/rstudio.sif Rscript analysis.R
```

Reproducibility Tips

1. **Document your environment**:
   ```
   # Save package versions
   apptainer exec /data/apps/rstudio.sif R -e 'sessionInfo()' > environment.txt
   ```
2. **Use specific image versions**:
   ```
   # Instead of "latest", use specific tags in definition files
   From: rocker/rstudio:4.3.0
   ```
3. **Version control your definition files**:
   ```
   git add rstudio_custom.def
   git commit -m "Added spatial analysis packages"
   ```
4. **Share your images**:
   - Export definition files to collaborators
   - Or share the `.sif` image file directly

Performance Considerations

Use appropriate resources:

```
# For memory-intensive tasks
#SBATCH --mem=32G

# For parallel processing
#SBATCH -c 8
```

Parallel R inside container:

```
library(parallel)
ncores <- as.numeric(Sys.getenv("SLURM_CPUS_PER_TASK", "1"))
cl <- makeCluster(ncores)
# Your parallel code here
stopCluster(cl)
```

Advanced Usage

Running Specific R Versions

Pull a specific R version:

```
Bootstrap: docker
From: rocker/r-ver:4.2.0

%post
    R --slave -e 'install.packages("your-packages")'
```

Python + R Environment

Create an environment with both Python and R:

```
Bootstrap: docker
From: rocker/rstudio:latest

%post
    # Install Python
    apt-get update
    apt-get install -y python3-pip python3-dev

    # Install Python packages
    pip3 install numpy pandas scikit-learn

    # Install reticulate for R-Python interface
    R --slave -e 'install.packages("reticulate")'
```

Use in R:

```
library(reticulate)
np <- import("numpy")
arr <- np$array(c(1, 2, 3, 4))
```

Using RStudio Server from Container

For interactive analysis through web browser (configured in OpenOnDemand):

```
# This is handled automatically by OpenOnDemand RStudio app
apptainer exec /data/apps/rstudio.sif rserver --www-port=8787
```

Installing Packages at Runtime

For quick testing (not persistent):

```
apptainer exec /data/apps/rstudio.sif R -e 'install.packages("newpackage", lib="/tmp/Rlibs")'
apptainer exec /data/apps/rstudio.sif R -e '.libPaths("/tmp/Rlibs"); library(newpackage)'
```

Note: This is temporary. For permanent installation, build a custom image.

Using renv for Package Management

Create reproducible R environments:

```
# In your project directory
library(renv)
renv::init()

# Install packages
install.packages(c("dplyr", "ggplot2"))

# Save state
renv::snapshot()

# Share renv.lock with collaborators
```

Run with container:

```
apptainer exec /data/apps/rstudio.sif R -e 'renv::restore()'
apptainer exec /data/apps/rstudio.sif Rscript analysis.R
```

Troubleshooting

Container Not Found

Error: `FATAL: container not found`

Solution:

```
# Check if image exists
ls -lh /data/apps/rstudio.sif

# For custom images
ls -lh $HOME/apps/rstudio_custom.sif

# Use full path
apptainer exec /data/apps/rstudio.sif Rscript script.R
```

Permission Denied

Error: `Permission denied` when reading/writing files

Solution:

- Ensure files are in accessible locations (home directory, current directory)
- Check file permissions: `ls -la yourfile.csv`
- Make files readable: `chmod 644 yourfile.csv`

Package Not Found

Error: `Error: package 'xyz' not found`

Solution:

1. Check if package is installed:

   ```
   apptainer exec /data/apps/rstudio.sif R -e 'installed.packages()[,"Package"]' | grep xyz
   ```

2. If not installed, build custom image with the package

3. Or install to user library (temporary):

```
install.packages("xyz", lib="~/R/library")
.libPaths("~/R/library")
```

Out of Memory

Error: `cannot allocate vector of size...`

Solution:

- Request more memory in SLURM:

```
#SBATCH --mem=64G
```

- Use memory-efficient R packages (`data.table` instead of `dplyr`)
- Process data in chunks
- Use `gc()` to free memory

Build Fails

Error: Build fails with various errors

Common solutions:

1. **Network issues**: Try again later
2. **Disk space**: Clean up old files in `~/.apptainer/cache`
3. **xattr errors**: Already fixed in the template script
4. **Package dependencies**: Add required system packages to definition file

Slow Performance

Causes:

- Not enough resources allocated
- Working with data on slow storage
- Inefficient R code

Solutions:

- Increase CPU/memory in SLURM directives
- Copy data to `/scratch` for faster I/O

- Profile and optimize your R code
- Use vectorized operations in R

Cannot Access Files

Problem: Files in other directories not accessible

Solution:

Bind additional directories:

```
export APPTAINER_BIND="/project:/project,/data:/data"
apptainer exec /data/apps/rstudio.sif Rscript script.R
```

Quick Reference

Common Commands

```
# Run R script
apptainer exec /data/apps/rstudio.sif Rscript script.R

# Interactive R
apptainer exec /data/apps/rstudio.sif R

# Shell inside container
apptainer shell /data/apps/rstudio.sif

# Check R version
apptainer exec /data/apps/rstudio.sif R --version

# List installed packages
apptainer exec /data/apps/rstudio.sif R -e 'installed.packages()[,1]'

# Run with custom image
apptainer exec $HOME/apps/rstudio_custom.sif Rscript script.R

# Bind additional directories
```

```
apptainer exec --bind /scratch:/scratch /data/apps/rstudio.sif Rscript script.R
```

SLURM Job Template

```
#!/bin/bash
#SBATCH --job-name=r_analysis
#SBATCH --time=02:00:00
#SBATCH --partition=normal
#SBATCH -c 4
#SBATCH --mem=16G
#SBATCH --output=%x-%j.out
#SBATCH --error=%x-%j.err

# Load modules if needed
# module load apptainer

# Navigate to working directory
cd $SLURM_SUBMIT_DIR

# Run R script with container
apptainer exec /data/apps/rstudio.sif Rscript analysis.R

# Or with custom image
# apptainer exec $HOME/apps/rstudio_custom.sif Rscript analysis.R
```

Definition File Template

```
Bootstrap: docker
From: rocker/rstudio:latest

%labels
    Author Your Name
    Version 1.0

%post
    # System packages
    apt-get update
```

```
    apt-get install -y package1 package2

    # R packages
    R --slave -e 'install.packages(c(
        "package1",
        "package2"
    ), repos="https://cloud.r-project.org/")'

%environment
    export LC_ALL=C
```

Resources

Documentation

- Apptainer Documentation: https://apptainer.org/docs/
- Rocker Project: https://rocker-project.org/
- R Package Documentation: https://cran.r-project.org/

Getting Help

- **Check installed packages**: `apptainer exec /data/apps/rstudio.sif R -e 'installed.packages()'`
- **Test interactively**: Use `apptainer shell` or `apptainer exec R` for testing
- **Review job logs**: Check `.out` and `.err` files in your job directory
- **Contact support**: Provide job ID and error messages

Example Workflow

1. **Develop interactively**:

   ```
   srun --pty --mem=8G -c 4 --time=01:00:00 bash
   apptainer exec /data/apps/rstudio.sif R
   # Test your code
   ```

2. **Create job script**:

```
vim my_job.sh
# Add SLURM directives and commands
```

3. **Submit and monitor**:

```
sbatch my_job.sh
squeue -u $USER
tail -f my_job-<jobid>.out
```

4. **Review results**:

```
less my_job-<jobid>.out
# Analyze output files
```

Last Updated: 2025-12-03

Questions? Contact VoWa HPC support team.

Revision #1
Created 2025-12-03 20:51:55 UTC by Stefan Hofstetter
Updated 2025-12-05 09:57:03 UTC by Stefan Hofstetter