OpenOnDemand Job Templates User Guide

This guide explains how to use job templates in OpenOnDemand to submit SLURM jobs efficiently.

What are Job Templates?

Job templates are pre-configured SLURM job scripts that help you quickly submit common types of jobs without writing scripts from scratch. Each template includes:

- Pre-configured SLURM directives (cores, memory, time limits)
- Example code or workflows
- Documentation and best practices
- Ready-to-run scripts

Accessing the Job Composer

1. **Log in to OpenOnDemand** at your cluster's URL
2. Click on **"Jobs"** in the top navigation menu
3. Select **"Job Composer"** from the dropdown

You'll see the Job Composer interface with:

- List of your existing jobs (left sidebar)
- Job details and files (main panel)
- Action buttons (Submit, Edit, Delete, etc.)

Creating Jobs from Templates

Step 1: Create New Job from Template

1. In the Job Composer, click **"New Job"** button
2. Select **"From Template"**
3. Choose a template from the list (see Available Templates)
4. Click **"Create New Job"**

The template will be copied to your jobs directory with all necessary files.

Step 2: Review Job Location

Your new job is created in:

```
~/ondemand/data/sys/myjobs/projects/default/<job-id>/
```

Each job gets a unique directory containing:

- `script.sh` - The SLURM job script
- Template-specific files (e.g., example R scripts, definition files)
- `README.md` - Documentation (if included)

Step 3: Understand the Job Structure

Every job template includes a `script.sh` file with SLURM directives at the top:

```
#!/bin/bash
#SBATCH --job-name=my_job        # Job name
#SBATCH --time=01:00:00          # Time limit (HH:MM:SS)
#SBATCH --partition=normal       # Queue/partition
#SBATCH -n 1                     # Number of tasks
#SBATCH -c 4                     # CPU cores per task
#SBATCH --mem=8G                 # Memory
#SBATCH --output=%x-%j.out       # Output file
#SBATCH --error=%x-%j.err        # Error file
```

Editing Job Scripts

Using the Built-in Editor

1. In Job Composer, select your job from the left sidebar
2. Click on `script.sh` in the file list
3. Click **"Edit"** button
4. Make your changes in the editor
5. Click **"Save"** when done

Common Edits

Change Resource Requirements

Edit the SLURM directives to match your needs:

```
#SBATCH --time=04:00:00          # Increase time limit
#SBATCH -c 8                     # Use more CPU cores
#SBATCH --mem=32G                # Request more memory
#SBATCH --partition=gpu          # Use GPU partition
```

Add Your Data Files

Edit the script to point to your actual data:

```
# Change this:
Rscript hello.R


# To this:
Rscript /path/to/your/analysis.R
```

Modify Job Name and Output

```
#SBATCH --job-name=my_analysis   # Descriptive name
#SBATCH --output=results_%j.out  # Custom output filename
```

Uploading Additional Files

1. In Job Composer, select your job
2. Click **"Open Dir"** to open the job directory in the file browser
3. Use **"Upload"** button to add your data files
4. Update the script to reference your uploaded files

Submitting Jobs

Submit Your Job

1. Select your job in the Job Composer
2. Review the script and ensure all settings are correct
3. Click the **"Submit"** button

You'll see a confirmation message with the job ID (e.g., "Job submitted successfully with ID: 12345")

What Happens Next?

1. **Queued**: Job enters the SLURM queue
2. **Running**: Job starts when resources are available
3. **Completed**: Job finishes (check output files for results)
4. **Failed**: Job encountered an error (check error file)

Monitoring Jobs

View Active Jobs

1. Click **"Jobs"** → **"Active Jobs"** in the top menu
2. You'll see all your running and pending jobs
3. Information displayed:
   - Job ID
   - Job Name
   - Status (Running, Pending, Completed, Failed)
   - Time elapsed
   - Nodes/cores used

Check Job Output

While the job is running or after completion:

1. In Job Composer, select your job
2. Click on the output file (e.g., `my_job-12345.out`)
3. Click **"View"** to see the contents
4. Click **"Refresh"** to update (for running jobs)

View Job Details

For detailed job information:

1. Go to **"Jobs"** → **"Active Jobs"**
2. Click on your job ID
3. View comprehensive details:
   - Start time
   - Resource usage

- Node assignment
- Full job parameters

Available Templates

Basic R Serial Job

Template: `rscript`

Purpose: Run R scripts on a single core

Includes:

- `script.sh` - SLURM job script
- `hello.R` - Example R script with system information

Use cases:

- Data analysis
- Statistical computing
- Report generation

How to customize:

1. Replace `hello.R` with your R script or upload your own
2. Edit `script.sh` to reference your script:

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

3. Adjust resources (memory, cores, time) as needed

Build Custom Apptainer Image

Template: `apptainer_builder`

Purpose: Build custom container images based on RStudio

Includes:

- `script.sh` - Build automation script
- `rstudio_custom.def` - Apptainer definition file
- `README.md` - Detailed instructions

Use cases:

- Installing additional R packages
- Adding system dependencies (GDAL, PROJ, etc.)
- Creating reproducible environments
- Custom software stacks

How to customize:

1. Edit `rstudio_custom.def` to add your packages:

```
%post
    apt-get update
    apt-get install -y your-system-packages

    R --slave -e 'install.packages(c("your", "packages"))'
```

2. Submit the job (build takes 1-4 hours)
3. Image is saved to `$HOME/apps/rstudio_custom.sif`
4. Use in future jobs or RStudio sessions

Advanced Usage

Creating Job Arrays

Run the same script multiple times with different parameters:

1. Edit your `script.sh` and add:

```
#SBATCH --array=1-10          # Run 10 instances
```

2. Use `$SLURM_ARRAY_TASK_ID` in your script:

```
Rscript analysis.R $SLURM_ARRAY_TASK_ID
```

Job Dependencies

Run jobs in sequence:

1. Submit first job and note the job ID (e.g., 12345)
2. Create second job with dependency:

```
#SBATCH --dependency=afterok:12345
```

Using Custom Container Images

After building a custom image:

1. Edit your job script
2. Change the container path:

```
# Instead of:
apptainer exec /data/apps/rstudio.sif Rscript script.R

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

Email Notifications

Get notified when jobs complete:

```
#SBATCH --mail-type=END,FAIL     # Email on end or failure
#SBATCH --mail-user=your.email@example.com
```

Using GPU Resources

For GPU-accelerated jobs:

```
#SBATCH --partition=gpu          # GPU partition
#SBATCH --gres=gpu:1             # Request 1 GPU
#SBATCH --gres=gpu:2             # Or request 2 GPUs
```

Parallel Processing in R

For multi-core R jobs:

```
#SBATCH -c 8                     # Request 8 cores
```

In your R script:

```
library(parallel)
library(doParallel)
```

```
# Use all available cores
n_cores <- as.numeric(Sys.getenv("SLURM_CPUS_PER_TASK"))
registerDoParallel(cores = n_cores)

# Your parallel code here
results <- foreach(i = 1:1000) %dopar% {
    # Computation
}
```

Troubleshooting

Job Stays in Pending State

Possible causes:

- Requested resources not available
- Partition full or not accessible
- Time limit too high for requested partition
- Account/QOS limits reached

Solutions:

1. Check active jobs: **"Jobs"** → **"Active Jobs"**
2. Reduce resource requests (cores, memory, time)
3. Try a different partition
4. Contact admin if issue persists

Job Fails Immediately

Check the error file (`*.err`):

1. In Job Composer, select your job
2. Open the `.err` file
3. Look for error messages

Common issues:

- File not found: Check paths are absolute or relative to job directory
- Permission denied: Ensure files are readable
- Module not loaded: Container may be missing dependencies
- Syntax errors: Review script for typos

Out of Memory Errors

Symptoms:

- Job fails with "out of memory" or "killed" message
- Exit code 137

Solutions:

1. Increase memory request:

   ```
   #SBATCH --mem=32G             # Instead of 8G
   ```

2. Use memory-efficient approaches in your code
3. Split job into smaller chunks

Container Image Not Found

Error: `FATAL: container not found`

Solutions:

1. Check the container path in your script
2. Verify the container exists:

   ```
   ls -l /data/apps/rstudio.sif
   ```

3. If using custom image, ensure build completed:

   ```
   ls -l $HOME/apps/rstudio_custom.sif
   ```

Job Takes Too Long

Options:

1. Request more time:

   ```
   #SBATCH --time=12:00:00
   ```

2. Optimize your code (vectorization, parallel processing)
3. Use more CPU cores if parallelizable
4. Check if you're in the correct partition for long jobs

Can't Edit Files

If editor doesn't work:

1. Click **"Open Dir"** in Job Composer
2. Use the file browser's built-in editor
3. Or download file, edit locally, re-upload

Need to Cancel a Job

1. Go to **"Jobs"** → **"Active Jobs"**
2. Find your job in the list
3. Click **"Delete"** or **"Cancel"** button
4. Confirm the cancellation

Best Practices

Resource Estimation

- **Start small**: Begin with minimal resources and increase if needed
- **Monitor usage**: Check actual resource usage after jobs complete
- **Be realistic**: Don't over-request resources (wastes queue time)

File Organization

- Keep related jobs in same project directory
- Use descriptive job names
- Document your workflow in README files
- Clean up old jobs periodically

Testing

- Test with small datasets first
- Use short time limits for testing
- Verify output before running large batches
- Use interactive sessions for debugging

Reproducibility

- Document software versions
- Use container images for consistent environments
- Save SLURM scripts with your results

- Note date and job ID in your analysis notes

Getting Help

Resources

- **Documentation**: Check README files in templates
- **Active Jobs**: Monitor job status and resource usage
- **Error Logs**: Always check `.err` files for failures

Contact Support

If you encounter persistent issues:

1. Note the job ID
2. Save error messages
3. Document what you've tried
4. Contact your HPC admin

Quick Reference

Common SLURM Directives

```
#SBATCH --job-name=name          # Job name
#SBATCH --partition=normal       # Queue/partition
#SBATCH --time=HH:MM:SS          # Time limit
#SBATCH -n 1                     # Number of tasks
#SBATCH -c 4                     # CPUs per task
#SBATCH --mem=8G                 # Memory per node
#SBATCH --mem-per-cpu=2G         # Memory per CPU
#SBATCH --output=file.out        # Output file
#SBATCH --error=file.err         # Error file
#SBATCH --mail-type=ALL          # Email notifications
#SBATCH --mail-user=email        # Email address
#SBATCH --gres=gpu:1             # GPU request
#SBATCH --array=1-10             # Job array
#SBATCH --dependency=afterok:123 # Job dependency
```

Environment Variables in Jobs

```
$SLURM_JOB_ID                  # Job ID
$SLURM_JOB_NAME                # Job name
$SLURM_SUBMIT_DIR              # Submission directory
$SLURM_JOB_NODELIST            # Assigned nodes
$SLURM_NTASKS                  # Number of tasks
$SLURM_CPUS_PER_TASK           # CPUs per task
$SLURM_ARRAY_TASK_ID           # Array index
```

Useful Commands (in scripts)

```
cd $SLURM_SUBMIT_DIR           # Go to submission directory
echo "Job ID: $SLURM_JOB_ID"   # Print job info
date                           # Timestamp
hostname                       # Node name
```

Example Workflow

Complete Example: Running an R Analysis

1. **Create job from template**:
   - Jobs → Job Composer → New Job → From Template
   - Select "Basic R Serial Job"
2. **Upload your R script**:
   - Click "Open Dir"
   - Upload your `analysis.R` file
3. **Edit the job script**:

```
#!/bin/bash
#SBATCH --job-name=my_analysis
#SBATCH --time=02:00:00
#SBATCH -c 4
#SBATCH --mem=16G

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

4. **Submit the job**:

- Click "Submit"
- Note the job ID

5. **Monitor progress**:
   - Jobs → Active Jobs
   - Check output file periodically
6. **Review results**:
   - Open `.out` file when job completes
   - Download output files if needed

Last Updated: 2025-12-03

Questions? Contact your SciIT team.

Revision #1
Created 2025-12-03 18:15:50 UTC by Stefan Hofstetter
Updated 2025-12-03 18:18:48 UTC by Stefan Hofstetter