Moab Docker Integration

Reference Guide for Moab 9.1 and Torque 6.1

November 2016

Adaptive
COMPUTING

Adaptive Computing Enterprises, Inc.
1712 S. East Bay Blvd., Suite 300
Provo, UT 84606
+1 (801) 717-3700
www.adaptivecomputing.com

Scan to open online help

Contents

Welcome

This documentation supports Adaptive Computing's Docker Integration package. Contact your Adaptive Computing account manager for more details.

Welcome to the Moab Docker Integration Reference Guide.

Docker, from Docker, Inc. allows you to package an application with all of its dependencies into a standardized unit for software development.

Moab and Torque now support Docker containers in which you can run your serial workloads. Docker containers provide an isolated environment with the correct linux distribution version of all the libraries the user needs to run the workload. System administrators can preset the containers or users can create their own images and have the administrator upload their images to a central registry so the users can create containers from them. You can also configure job templates to force workloads and/or users to run inside Docker containers, as well as running preemptible or interactive jobs in containerized environments.

The following chapters will help you quickly get started:

- Chapter 1 Introduction to Moab Docker Integration on page 3 – Provides a brief introduction to using Docker containers. Also includes an illustration of the Docker container lifecycle.
- Chapter 2 Installation and Configuration on page 5 – Contains requirements information and procedures to set up Moab and Torque for Docker support. Also includes instructions for copying the Docker job start scripts.
- Chapter 3 Docker Job Submission on page 19 – Provides instructions and information on how to submit jobs using Docker containers.
- Chapter 4 Troubleshooting on page 25 – Provides information useful when troubleshooting your Moab Docker Integration.

Chapter 1 Introduction to Moab Docker Integration

This chapter provides information about Moab Docker Integration.

In this chapter:

About Moab Docker Integration

Moab Docker Integration (also referred to as Docker support) is an additional package you can elect to include with your Moab configuration. This package enables Moab and Torque jobs to optionally run in a Linux container using Docker. Jobs may be run in a container of the user's choice and, within the container, the job script is run under the submitting user id. A container job may also be preempted (checkpointed at the container level; processes are not checkpointed) and resumed in the same container in which it was previously run.

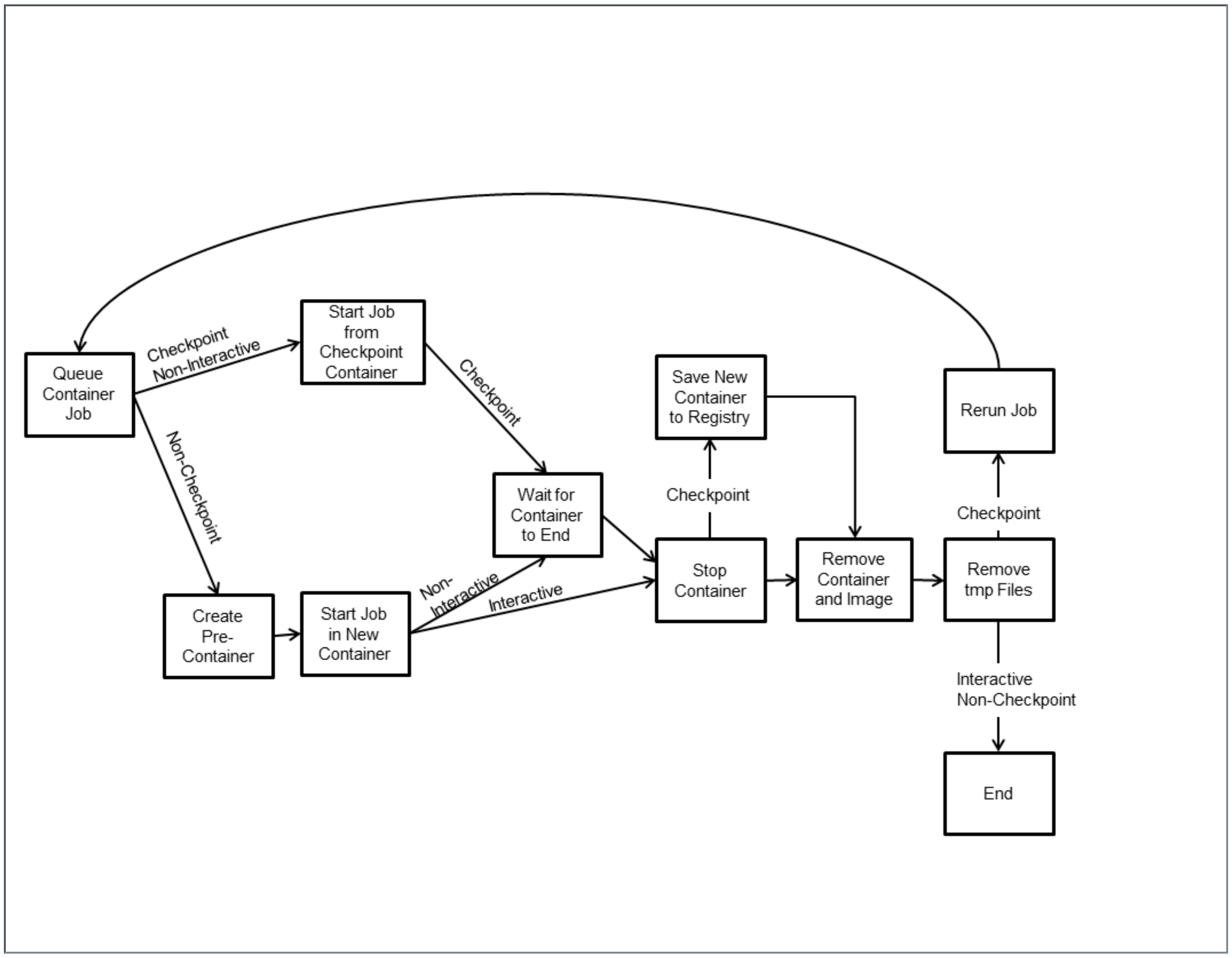
Specifically, this Reference Guide guide will show setting up a CentOS5 container to run jobs on your CentOS 7 or SLES 12 Host OS.

Docker Integration requires Torque Resource Manager as the Moab Resource Manager.

Docker Integration is available for Moab 9.1 and Torque 6.1.

Container Job Lifecycle

The following image shows the job lifecycle when using Docker containers.


Queue Container Job
Checkpoint Non-Interactive
Start Job from Checkpoint Container
Checkpoint
Non-Checkpoint
Create Pre-Container
Start Job in New Container
Non-Interactive
Interactive
Wait for Container to End
Save New Container to Registry
Checkpoint
Stop Container
Remove Container and Image
Rerun Job
Checkpoint
Remove tmp Files
Interactive Non-Checkpoint
End

Chapter 2 Installation and Configuration

This chapter provides information on how to set up and configure Torque Resource Manager for Docker container support.

This chapter assumes you have already installed Moab Workload Manager and Torque Resource Manager; including the Torque MOM Hosts (also referred to as compute nodes).

In this chapter:

Requirements

This topic contains information on the requirements for the Moab Server Host and the Torque MOM Hosts (also referred to as the compute nodes).

Moab Server Host

This is the host on which Moab Server resides.

Requirements:

- Moab Workload Manager 9.1

Torque MOM Hosts

These are the hosts on which the Torque MOM Client reside (also referred to as the compute nodes).

Requirements:

- Red Hat 7-based or SUSE 12-based Host OS

  For Red Hat 7-based systems, Security Enhanced Linux (SELinux) may be in any mode (enforcing, permissive, or disabled).

- Python 2.7; assumed to be installed with the Host OS
- Docker 1.8.2 and after

- Torque 6.1.0 and after
- If using preemption, all nodes should have a shared $HOME file system

Installing and Configuring Docker

This topic provides instructions on installing and configuring Docker as part of your Moab Docker Integration. You should install and configure Docker on each Torque MOM Host.

> If preemption is part of your system configuration, you will also need to repeat these steps on the host on which your local registry will reside. See also Setting Up the Local Registry (Preemption Only) on page 16.

In this topic:

- Supported OS and Docker Versions on page 6
- Install and Configure Docker on CentOS 7 on page 7
- Install and Configure Docker on RHEL 7 on page 10
- Install and Configure Docker on SLES 12 on page 12

Supported OS and Docker Versions

This topic provides steps for installing and configuring Docker on the supported combinations of operating systems and Docker versions, as shown in the table below.

Operating System	Docker Version
Red Hat-based systems	
CentOS 7 – Default OS Install	1.10.3
CentOS 7 – Yum Install	1.12.1
CentOS 7 – Script Install	1.12.1
Red Hat Enterprise Linux (RHEL) 7	1.12.1
SUSE-based systems	
SUSE Linux Enterprise Server (SLES) 12	1.9.1

Docker is not supported for Scientific Linux.

Install and Configure Docker on CentOS 7

There are three methods for installing Docker on CentOS 7. The default OS install method installs Docker 1.10.3. The yum and script install methods install Docker 1.12.1 from the Docker repository. See https://docs.docker.com/engine/installation/linux/centos/.

- Default OS Install (for Docker 1.10.3)
- Yum Install (for Docker 1.12.1)
- Script Install (for Docker 1.12.1)

Default OS Install (for Docker 1.10.3)

Do the following:

1. Make sure existing yum packages are up-to-date.
   ```
   yum update
   ```
2. Install the Docker package.
   ```
   yum install docker
   ```
3. Edit `/etc/sysconfig/docker` by replacing the existing OPTIONS line with the following.
   ```
   OPTIONS='-s devicemapper --storage-opt dm.fs=xfs --exec-opt
   native.cgroupdriver=cgroupfs'
   ```
4. Start/restart Docker.
   ```
   systemctl restart docker.service
   ```
5. Set Docker to start after reboot.
   ```
   systemctl enable docker.service
   ```
6. Verify the installation.
   ```
   docker run hello-world
   ```
7. Download any Docker images you want to make available to users on your system. For example on each MOM host:
   ```
   docker pull centos:6
   ```

Yum Install (for Docker 1.12.1)

Do the following:

1. Make sure existing yum packages are up-to-date.

```
yum update
```

2. Add the yum repo.

```
tee /etc/yum.repos.d/docker.repo <<-EOF
[dockerrepo]
name=Docker Repository
baseurl=https://yum.dockerproject.org/repo/main/centos/7/
enabled=1
gpgcheck=1
gpgkey=https://yum.dockerproject.org/gpg
EOF
```

3. Install the Docker package.

```
yum install docker-engine
```

4. Create the service directory.

```
mkdir /etc/systemd/system/docker.service.d
```

5. In the directory you just created, create the `execstart_override.conf` file and add "[Service]" and "ExecStart=" as the first and second lines, respectively.
6. From `/lib/systemd/system/docker.service`, copy the ExecStart line and place it as the last line in the `execstart_override.conf` file.
7. In the `execstart_override.conf` file, append the last line to include `-s devicemapper --storage-opt dm.fs=xfs --exec-opt native.cgroupdriver=cgroupfs`.

   The following is an example of the configured `execstart_override.conf` file for Docker 1.12.1.

```
[Service]
ExecStart=
ExecStart=/usr/bin/dockerd -s devicemapper --storage-opt dm.fs=xfs --exec-opt
native.cgroupdriver=cgroupfs
```

8. Reload the systemd manager configuration.

```
systemctl daemon-reload
```

9. Start/restart Docker.

```
systemctl restart docker.service
```

10. Set Docker to start after reboot.

```
systemctl enable docker.service
```

11. Verify the installation.

```
docker run hello-world
```

12. Download any Docker images you want to make available to users on your system. For example on each MOM host:

```
docker pull centos:6
```

Script Install (for Docker 1.12.1)

Do the following:

1. Make sure existing yum packages are up-to-date.

```
yum update
```

2. Run the Docker installation script.

```
curl -fsSL https://get.docker.com/ | sh
```

3. Create the service directory.

```
mkdir /etc/systemd/system/docker.service.d
```

4. In the directory you just created, create the `execstart_override.conf` file and add "[Service]" and "ExecStart=" as the first and second lines, respectively.
5. From `/lib/systemd/system/docker.service`, copy the ExecStart line and place it as the last line in the `execstart_override.conf` file.
6. In the `execstart_override.conf` file, append the last line to include `-s devicemapper --storage-opt dm.fs=xfs --exec-opt native.cgroupdriver=cgroupfs`.

   The following is an example of the configured `execstart_override.conf` file for Docker 1.12.1.

```
[Service]
ExecStart=
ExecStart=/usr/bin/dockerd -s devicemapper --storage-opt dm.fs=xfs --exec-opt
native.cgroupdriver=cgroupfs
```

7. Reload the systemd manager configuration.

```
systemctl daemon-reload
```

8. Start/restart Docker.

```
systemctl restart docker.service
```

9. Set Docker to start after reboot.

```
systemctl enable docker.service
```

10. Verify the installation.

```
docker run hello-world
```

11. Download any Docker images you want to make available to users on your system. For example on each MOM host:

```
docker pull centos:6
```

Install and Configure Docker on RHEL 7

There are two methods for installing Docker on RHEL 7, yum install and script install. Both methods install Docker 1.12.1 from the Docker repository. See https://docs.docker.com/engine/installation/linux/rhel/.

- Yum Install (for Docker 1.12.1)
- Script Install (for Docker 1.12.1)

Yum Install (for Docker 1.12.1)

Do the following:

1. Make sure existing yum packages are up-to-date.

```
yum update
```

2. Add the yum repo.

```
tee /etc/yum.repos.d/docker.repo <<-EOF
[dockerrepo]
name=Docker Repository
baseurl=https://yum.dockerproject.org/repo/main/centos/7
enabled=1
gpgcheck=1
gpgkey=https://yum.dockerproject.org/gpg
EOF
```

3. Install the Docker package.

```
yum install docker-engine
```

4. Create the service directory.

```
mkdir /etc/systemd/system/docker.service.d
```

5. In the directory you just created, create the `execstart_override.conf` file and add "[Service]" and "ExecStart=" as the first and second lines, respectively.

6. From `/lib/systemd/system/docker.service`, copy the ExecStart line and place it as the last line in the `execstart_override.conf` file.
7. In the `execstart_override.conf` file, append the last line to include `-s devicemapper --storage-opt dm.fs=xfs --exec-opt native.cgroupdriver=cgroupfs`.

   The following is an example of the configured `execstart_override.conf` file for Docker 1.12.1.

   ```
   [Service]
   ExecStart=
   ExecStart=/usr/bin/dockerd -s devicemapper --storage-opt dm.fs=xfs --exec-opt
   native.cgroupdriver=cgroupfs
   ```
8. Reload the systemd manager configuration.

   ```
   systemctl daemon-reload
   ```
9. Start/restart Docker.

   ```
   systemctl restart docker.service
   ```
10. Set Docker to start after reboot.

    ```
    systemctl enable docker.service
    ```
11. Verify the installation.

    ```
    docker run hello-world
    ```
12. Download any Docker images you want to make available to users on your system. For example on each MOM host:

    ```
    docker pull centos:6
    ```

Script Install (for Docker 1.12.1)

Do the following:

1. Make sure existing yum packages are up-to-date.

   ```
   yum update
   ```
2. Run the Docker installation script.

   ```
   curl -fsSL https://get.docker.com/ | sh
   ```
3. Create the service directory.

   ```
   mkdir /etc/systemd/system/docker.service.d
   ```

4. In the directory you just created, create the `execstart_override.conf` file and add "[Service]" and "ExecStart=" as the first and second lines, respectively.
5. From `/lib/systemd/system/docker.service`, copy the ExecStart line and place it as the last line in the `execstart_override.conf` file.
6. In the `execstart_override.conf` file, append the last line to include `-s devicemapper --storage-opt dm.fs=xfs --exec-opt native.cgroupdriver=cgroupfs`.

   The following is an example of the configured `execstart_override.conf` file for Docker 1.12.1.

   ```
   [Service]
   ExecStart=
   ExecStart=/usr/bin/dockerd -s devicemapper --storage-opt dm.fs=xfs --exec-opt
   native.cgroupdriver=cgroupfs
   ```
7. Reload the systemd manager configuration.

   ```
   systemctl daemon-reload
   ```
8. Start/restart Docker.

   ```
   systemctl restart docker.service
   ```
9. Set Docker to start after reboot.

   ```
   systemctl enable docker.service
   ```
10. Verify the installation.

    ```
    docker run hello-world
    ```
11. Download any Docker images you want to make available to users on your system. For example on each MOM host:

    ```
    docker pull centos:6
    ```

Install and Configure Docker on SLES 12

The latest supported Docker packages are inside the Container module. Do the following:

1. Access the Docker packages by choosing one of these options.
   a. Enable the Container module.
      i. Start YaST and select *Software > Software Repositories*.
      ii. Click *Add* to open the add-on dialog.
      iii. Select *Extensions and Module from Registration Server* and click *Next*.

iv. From the list of available extensions and modules, select *Container Module* and click *Next*. The Container module and its repositories are added to your system.

v. If you use Subscription Management Tool, update the list of repositories at the SMT server.

b. Execute the following command:

```
SUSEConnect -p sle-module-containers/12/x86_64 -r ''
```

The `''` following the `-r` flag above is two single quotes. It is necessary to avoid a limitation of SUSEConnect.

2. Run the Docker installation script.

```
zypper install docker
```

3. Edit `/etc/sysconfig/docker` by replacing the existing DOCKER_OPTS line with the following.

```
DOCKER_OPTS='-s devicemapper --storage-opt dm.fs=xfs --exec-opt
native.cgroupdriver=cgroupfs'
```

4. Start/restart Docker.

```
systemctl restart docker.service
```

5. Set Docker to start after reboot.

```
systemctl enable docker.service
```

6. Verify the installation.

```
docker run hello-world
```

7. Download any Docker images you want to make available to users on your system. For example on each MOM host:

```
docker pull centos:6
```

Configuring Torque

This topic provides additional configuration requirements for your Torque Server Host and Torque MOM Hosts as part of your Moab Docker Integration.

You must have installed Moab Workload Manager and Torque Resource Manager; including the Torque MOM Hosts (also referred to as compute nodes) before proceeding with this topic.

In this topic:

Configure the Torque Server Host

On the Torque Server Host, do the following:

1. Set root to be a queue manager on each Torque MOM Host.

```
qmgr -c "s s managers += root@<MOM Host>"
```

2. Set kill delay to 30 seconds.

```
qmgr -c "s s kill_delay=30"
```

> If you have a slow file system (or network) and the Docker images are taking awhile to clean up, you may need to increase kill_delay to a higher value to give the job starter enough time to cleanup after the container has been terminated.

Configure the Torque MOM Hosts

> The following instructions assume your Torque home directory is `/var/spool/torque` on each Torque MOM Host. If your home directories are different, you will need to adjust the instructions in this topic accordingly.

On *each* Torque MOM Host, do the following:

1. Install and enable cgroups. See Torque NUMA-Aware Configuration in the *Torque Resource Manager Administrator Guide* for detailed instructions.
2. Configure Torque to run the job starter with elevated privileges. Append these lines to `/var/spool/torque/mom_priv/config`:
   - $job_starter /var/spool/torque/mom_priv/job_starter
   - $job_starter_run_privileged true
   - $spool_as_final_name true (required only if using preemption)
3. Restart pbs_mom.

```
systemctl restart pbs_mom.service
```

Installing and Configuring Docker Job Start Scripts

This topic provides instructions on how to install and configure the Docker job start scripts.

Install Docker Job Scripts

On each Torque MOM Host, do the following:

1. Download the job scripts from Adaptive Computing (http://www.adaptivecomputing.com/support/download-center/docker/).

   The job scripts require a valid login name and password to access. Contact your Adaptive Computing account manager if you do not have your access information, or have other issues downloading these scripts.

2. Copy these job starter scripts to `/var/spool/torque/mom_priv` after customization listed in the following steps:
   - job_starter
   - epilogue
   - job_starter_common.pyc
   - job_starter_config.py
   - job_starter_mountpoints.json (optional)
3. Customize the "job_starter_config.py" file. A sample script is provided; edit as needed.
4. If you installed "job_starter_mountpoints.json", configure the container mount points. For example:

```
# cat job_starter_mountpoints.json
{
      "*all*": ["/tmp/:/tmp/:rw", "$HOME:$HOME:rw"],
      "centos:5": ["/var/tmp:/var/tmp:ro"]
}
```

   In this example, all containers will have the Docker host's /tmp and the user's home directory mounted with read/write privileges inside the container, and when a user requests a centos:5 image, /var/tmp from the Docker host will be mounted in /var/tmp in the container in read only mode.

Setting Up the Local Registry (Preemption Only)

If preemption is part of your configuration, you will need to set up the local registry.

Preemption is only available for single-node jobs.

Set Up the Local Registry

On the local registry host *and* on each Torque MOM Host, configure Docker using the method for your Docker version and distro.

In the following instructions, "myreg.host.com" is used as the local registry name. Change this to match your local registry information.

Configure Docker 1.10.3 on CentOS 7

Do the following:

1. Edit the `/etc/systemd/docker` file to uncomment the `INSECURE_REGISTRY` line and add the registry host.

```
INSECURE_REGISTRY='--insecure-registry myreg.host.com:5000'
```

2. Restart Docker.

```
systemctl restart docker.service
```

3. On the local registry host, do the following:
   a. Set up the local registry. Complete the steps in https://docs.docker.com/registry/deploying/.
   b. Start the registry.

   ```
   docker run -d -p 5000:5000 --restart=always --name registry registry:2
   ```

Configure Docker 1.12.1 on RHEL 7 or CentOS 7

Do the following:

1. Edit `/etc/systemd/system/docker.service.d/execstart_override.conf` to append `--insecure-registry myreg.host.com:5000` to the last ExecStart line.
2. Reload the `systemd` manager configuration.

```
systemctl daemon-reload
```

3. Restart Docker.

```
systemctl restart docker.service
```

4. On the local registry host, do the following:
   a. Set up the local registry. Complete the steps in https://docs.docker.com/registry/deploying/.
   b. Start the registry.

   ```
   docker run -d -p 5000:5000 --restart=always --name registry registry:2
   ```

Configure Docker 1.9.1 on SLES 12

Do the following:

1. Edit `/etc/sysconfig/docker` to add `--insecure-registry myreg.host.com:5000` to the DOCKER_OPTS value.
2. Restart Docker.

```
systemctl restart docker.service
```

3. On the local registry host, do the following:
   a. Set up the local registry. Complete the steps in https://docs.docker.com/registry/deploying/.
   b. Start the registry.

   ```
   docker run -d -p 5000:5000 --restart=always --name registry registry:2
   ```

Chapter 3 Docker Job Submission

The Moab Docker Integration component provides several options for submitting jobs.

You can submit interactive, non-interactive, or preemptible jobs. You can also choose whether to submit those jobs directly to Torque or by using a Moab job template.

In this chapter:

Related Topics

Container Job Lifecycle on page 4

Submitting Interactive and Non-Interactive Jobs

This topic provides instructions for submitting interactive and non-interactive jobs.

Submit Interactive Job

Use the following commands to submit an interactive job from Moab or Torque.

- Torque

```
qsub -I -v PBS_CONTAINERINFO=<imagename> job.pbs
```

- Moab

```
msub -I -v PBS_CONTAINERINFO=<imagename> job.pbs
```

Submit Non-Interactive Job

Use the following commands to submit a non-interactive job from Moab or Torque.

- Torque

```
qsub -v PBS_CONTAINERINFO=<imagename> job.pbs
```

- Moab

```
msub -v PBS_CONTAINERINFO=<imagename> job.pbs
```

Terminating Jobs

This topic provides instructions for terminating jobs.

Terminate a Job

- Torque

```
qdel <jobid>
```

- Moab
  - `mjobctl -N TERM <jobid>` is the preferred method for terminating Docker jobs. This command removes both non-preemptible and preemptible (checkpointed) Docker jobs.
  - `mjobctl -c <jobid>` or `canceljob <jobid>`, when used with preemptible (checkpointed) Docker jobs, sends SIGUSR1 to the Docker job causing it to checkpoint. For non-preemptible Docker jobs, these commands terminate Docker jobs similar to how these commands terminate non-Docker jobs.

Using Templates for Docker Jobs

You can set up job templates in Moab to automatically push the PBS_CONTAINERINFO environment variable for your users and to set up different user policies in Moab.

Create a Template

On the Moab Server Host, add the template information to moab.cfg. The following example creates a template for Docker jobs using a CentOS 5 container.

```
# CentOS 5
JOBCFG[centos5template] SELECT=TRUE
JOBCFG[centos5template] FLAGS=RESTARTABLE
JOBCFG[centos5template] ENV=PBS_CONTAINERINFO=localhost:5000/centos:5
JOBCFG[centos5template] ENV=PBS_CONTAINERHOSTNAME=centos5
```

Submit Jobs Using a Template

On the Moab Server Host, do the following:

```
msub -l template=centos5template myjob.sh
```

Enabling Preemptible Docker Jobs

This topic provides information and instructions on how to enable preemptible Docker jobs.

In this topic:

Configure Moab to Preempt Docker Jobs

On the Moab Server Host, do the following:

1. Edit the `moab.cfg` file to add the checkpoint parameters.

```
# Signal to send to RM when job is checkpointed
RMCFG[pbs] CHECKPOINTSIG=SIGUSR1

# How long to wait for a job to checkpoint before canceling it
RMCFG[pbs] CHECKPOINTTIMEOUT=1:00
GUARANTEEDPREEMPTION TRUE
PREEMPTPOLICY CHECKPOINT
```

2. Create a new or use an existing job template for preemption and then edit the `moab.cfg` file to make that template restartable (all jobs submitted using this template can be restarted). Add this line (where [centos5template] is your template name):

```
JOBCFG[centos5template] FLAGS=RESTARTABLE
```

See Using Templates for Docker Jobs on page 20 for additional information on templates.

3. Restart Moab

```
[root]# systemctl restart moab.service
```

Communicate With the Local Registry

In the following instructions, "myrepo.host.com" is used as the local registry name. Change this to match your local registry information.

Do the following:

1. If you have not already done so, set up the local registry. See Setting Up the Local Registry (Preemption Only) on page 16.
2. On each Torque MOM Host, edit the REGISTRY_URL parameter in the job_starter_config.py script to point to the local registry. The REGISTRY_URL is the host name of the local registry host and the port number. For example:

```
REGISTRY_URL='myrepo.host.com:5000'
```

Request a Preemptible Docker Job

Once the local registry, a template, and the moab.cfg file are configured for preemptible Docker jobs, you only need to submit the job using the template configured for preemption (centos5template is used in this documentation).

Test Docker Job Preemption

This section contains instructions on how to test preemption for a Docker job. In these instructions you will modify the job's checkpoint values and submit the job using a restartable job template (centos5template).

On the Moab Server Host, do the following:

1. Submit a job script that records its progress and can recover from where it left off. For example:

```
cat sample-checkpoint.sh
#!/bin/bash
# where to store our progress
CHECKPOINT_FILE=/var/tmp/checkpoint.dat
# how many steps do you want to iterate through?
NSTEPS=20
# how many seconds to sleep for in each step
SLEEP_AMOUNT=5

START=1
if test -f $CHECKPOINT_FILE
then
        echo "Checkpoint data found! Resuming execution..."
        START=`cat $CHECKPOINT_FILE`
fi
echo "Starting from step $START..."
while test $START -lt `expr $NSTEPS + 1`
do
        echo "I'm in step $START"
        # increment counter
        START=`expr $START + 1`
        # sleep for $SLEEP_AMOUNT seconds
        sleep $SLEEP_AMOUNT
        # log progress...
        echo $START > $CHECKPOINT_FILE
done
```

2. Edit the variables that control the application's behavior.

   In the previous step, you will find these variables that can control the application's behavior:

   - CHECKPOINT_FILE
   - NSTEPS
   - SLEEP_AMOUNT

   Modify those variables for your test. In this example, the job will iterate through 20 different steps, sleeping 5 seconds between each of the steps and recording its progress to a checkpoint.dat file. The other variables do not need to be changed.

3. Submit the job using a template that is configured for preemption. For example:

```
msub -l template=centos5template sample-checkpoint.sh

142
```

   This will submit the job requesting the centos5 template. After the job has been running for a few seconds, if you ask Moab to checkpoint it, Moab will push a SIGUSR1 signal to your job and it will then archive a copy of your container's file system in the central registry. The next time it restarts the job, it can pick up where it left off.

4. Wait a few seconds to give the job time to run and then preempt it:

```
mjobctl -C 142

job 142 successfully preempted
```

5. Wait a few more seconds and then you should see the job getting requeued and restarted.

6. After the job has finished, review its output file. The output file will show that the job was able to pick up where it left off.

```
cat sample-checkpoint.sh.o142
Starting from step 1...
I'm in step 1
I'm in step 2
I'm in step 3
I'm in step 4
I'm in step 5
I'm in step 6
Checkpoint data found! Resuming execution...
Starting from step 6...
I'm in step 6
I'm in step 7
I'm in step 8
I'm in step 9
I'm in step 10
I'm in step 11
I'm in step 12
I'm in step 13
I'm in step 14
I'm in step 15
I'm in step 16
I'm in step 17
I'm in step 18
I'm in step 19
I'm in step 20
```

Related Topics

Using Templates for Docker Jobs on page 20

Chapter 4 Troubleshooting

This chapter provides information useful when troubleshooting yourMoab Docker Integration.

Also see Container Job Lifecycle on page 4 for detailed information about the job lifecycle.

In this chapter:

- Known Issues or Limitations on page 25

Known Issues or Limitations

This topic lists current known issues (bugs) and limitations (not supported functionality).

Known Issues

- "Docker logs --follow" command has a known issue where it may repeat the first line of output. As a result, non-interactive jobs may log the first line of output to stdout twice.
- For a slow file system (network), Docker images may take awhile to clean up. As part of the initial Moab Docker Integration, a kill delay setting of 30 seconds is specified. Increase this value to give the job starter enough time to clean up the container.

```
qmgr -c "s s kill_delay=<integer>"
```

- A SIGKILL sent to the job without a SIGTERM sent first *and* without enough time after the SIGTERM to handle the container cleanup, can result in any or all of the following:
    - container left running after batch job
    - container image not cleaned up
    - temporary files "/tmp/<*jobid*>" not removed

Limitations

- Only serial jobs are supported at this time; no support for parallel jobs
- Interactive jobs cannot be checkpointed.
- Job prologue/epilogue scripts run outside container job.

- Moab "mjobctl -c <*jobid*>" or "canceljob <*jobid*>", when used with preemptible (checkpointed) Docker jobs, sends SIGUSR1 to the Docker job causing it to checkpoint. See Terminating Jobs on page 20 for additional information on terminating preemptible and non-preemptible Docker jobs.
- Cluster environments of mixed Red Hat 7 -based and SUSE 12 -based hosts running Docker has not been tested.