EVIDEN
an atos business

# QuickStart configuration for Qaptiva Slurm

## Introduction

By default, Qaptiva Power Access uses a simple FIFO queue to manage jobs. This queue is implemented using a PostgreSQL database. Qaptiva Slurm enables Qaptiva Power Access to use the Slurm scheduler. The Slurm scheduler does not replace the default scheduler, it is used ahead of it (two-scheduler method).

## Requirement

A functional Qaptiva Appliance / Qaptiva Power Access (aka QLM/QLMaaS) installation at level 1.10.2 or above and a functional Slurm cluster at level 23.02.2 or above.

This QuickStart considers the following systems:

- A Qaptiva 800s Appliance
- A Qaptiva Power Access Proxy (aka QLMaaS Front end/proxy)
- A Slurm login node on RHEL 9 / Rocky 9
- A Slurm controller node
- A Slurm compute node on RHEL 9 / Rocky 9
    - This node serves as a Slurm proxy to Qaptiva; as such it is also referred to as qlmproxy.
    - One per Appliance to be proxied and dedicated to being used by Qaptiva.
- MyQLM installation [optional for testing but in the real world]

Any of these systems can be bundled together according to the site requirements. For example, Qaptiva Power Access Proxy can be on the Qaptiva Appliance, as well as all Qaptiva Slurm components. See Appendix A for a workflow schematic with all components distributed across different systems.

## Licensing

A QLMaaS license (/etc/qlmaas/license) must be present and be allowed to replace the default scheduler (i.e. the QLMaaS license must not be a limited edition) . Please ask your Eviden support person to deliver an appropriate license.

## Qaptiva Bundle

Starting with Qaptiva 1.10, the Qaptiva Installer prompts to install Qaptiva Slurm on the Appliance itself. This is made for sites that would like to beneficiate of the Slurm scheduling added values such as fair share queuing without having a full Slurm Cluster or open the Appliance to one. In this scenario, the Qaptiva installer will install the Qaptiva Slurm RPMS on the Appliance, along with the Slurm product and its main dependencies (i.e. Munge and MariaDB). A simple Slurm configuration self-contained as a single node (the Appliance itself) is made by the installer. If this is your context, you have nothing more to install. You do not need to read the installation chapter.

However, the installer will not automatically switch the default scheduler to use Slurm. See the configuration chapter below if you come to this QuickStart to enable the Slurm Scheduler in the context of the Qaptiva bundle.

## Installation prerequisite

This QuickStart does not provide information on how to install Qaptiva, or a Slurm Cluster. To install Qaptiva/Qaptiva Power Access, please refer to the document 'QLM_Installation_Instructions-1.10.2.pdf.'

The product distribution can be found on OBAN, the official Eviden support server.

## Structure on OBAN

```
tree …/qaptiva-slurm/0.1.2
├── doc
│   └── Qaptiva_Slurm_QuickStart-0.1.2.pdf
├── fixes
├── md5sum
└── qaptiva-slurm-installer.tar
```

Download the qaptiva-slurm-installer.tar file above under /tmp of the participating systems. Then, as root, extract the installer.

```
tar xf /tmp/qaptiva-slurm-installer.tar -C /var
└── qaptiva-slurm-installer
    ├── others
    │   └── qaptiva-slurm.repo
    ├── rpms
    │   └── qat
    |       rpm files…
    │       └── repodata
    |           index files…
    └── support
        ├── ref310.tgz
        ├── ref311.tgz
        ├── ref312.tgz
        ├── ref39.tgz
        └── thrift
            ├── six-1.16.0-py2.py3-none-any.whl
            ├── thrift-0.20.0.tar.gz
            └── thrift-c++-0.20.0.tgz
```

## YUM repo file

Copy the repo file into yum.repos.d.

```
cp /var/qaptiva-slurm-installer/others/qaptiva-slurm.repo /etc/yum.repos.d/
```

## Installation

The software's core functionalities are packaged as a set of RPMs, along with a few supporting tarball files; they are meant to be installed on each participating system depending on the functions(s) performed by each of those systems.

The RPMS are grouped by the function that they are made to perform:

- slurm-midtier for the system that hosts the Power Access Front-End services,
- slurm-loginnode for the system that performs the login node functionality in the Slurm Cluster,
- slurm-qlmproxy for the Slurm compute node chosen to act as a proxy to the Qaptiva Appliance.

Installing any of those will automatically install their dependencies.

The three groups above can be installed on the same system if all functions are concentrated on a single system (i.e. the Appliance), and up to four systems when the functions are fully distributed (i.e. the Appliance, a separate proxy system to run the Power Access Front-End services, a login node, and a compute node to act as a proxy for the Appliance); and of course any combinations in between. The Slurm controller node is not part of the discussion here as we do not need to install anything on it (with the exception if the functionality is shared within another node in which we do install our software).

Additionally, the RPMs are categorized based on the Python version they support. Qaptiva Slurm supports the following python versions: 39, 310, 311 and 312. For instance, for the slurm-midtier package group, we will have the following RPMS:

- qlmaas39-slurm-midtier
- qlmaas310-slurm-midtier
- qlmaas311-slurm-midtier
- qlmaas312-slurm-midtier

Practically, the only groups required are the ones with Python 3.12; please read below to understand why.

## Install Python base versions

On the Qaptiva systems (i.e. Appliance and Qaptiva Power Access Proxy), all supported base python versions are installed by the Qaptiva installer. There is nothing more to do. On those system, the command pyversion can be used to list the available versions.

However, on the other systems (i.e. separate login node and compute node), we do not provide base python versions.

Please note that the different python versions on the Qaptiva systems allows users to use the versions of their chosen when using Jupyter Notebooks or when running command line programs, but that all QLMaaS and QLMaaS Slurm services are preset to use python 3.12.

*Consequently, python 3.12 is the only required base version to be available on all systems.*

## Install Python Reference Virtual Environments

Having python 3.12 operational as seen above is not enough, support python packages must also be present.

We provide tarball files that contain the necessary support packages. Extracting those tarballs creates what we call the Python Reference Virtual Environments.

On the Qaptiva systems, like for the base python versions, those would have already been installed by the Qaptiva installer and there is nothing further to do. You can use the command workon -lr to list the ones available.

If there are other systems involved (i.e. separate login node and compute node), you must install our Reference Environment for python 3.12. Please see below for instructions on how to install it.

```
# Install the reference virtual environment for python 3.12
mkdir -p /usr/local/qaptiva/python_references
tar xfz /var/qaptiva-slurm-installer/support/ref312.tgz -C /usr/local/qaptiva/python_references
```

## Install Thrift

Additionally, all Qaptiva and Qaptiva Slurm services communicates using the Apache Thrift API. The API supports both python and C++, which are the two interfaces used in Qaptiva. This adds the constraint that all participating systems must have Thrift python and C++ installed. Furthermore, because of the dependance between objects serialization and deserialization, one of the main functions performed by Thrift, all thrift versions must be compatible, and more likely be the same.

Because Qaptiva runs Thrift 0.20, we highly recommend having Thrift 0.20 on all participating systems. Other combinations have not been tested and will likely fail.

### Thrift Python part

The python side is already part of the Reference Virtual Environments discussed above, and there is nothing to do on any of the systems.

### Thrift C++ part

Practically, RHEL/EPEL 9 repos come with Thrift 0.16, and there are no RPMS for Thrift 02.0 at the time of this writing. Consequently, we provide tarballs files containing Thrift 0.20.

On the Qaptiva systems, thrift would have already been installed by the Qaptiva installer and there is nothing further to do in this matter.

On the other systems involved, you must uninstall Thrift 0.16 if the RPMS are installed, then extract our tarball.

```
# Install thrift 0.20
dnf remove $(rpm -qa | grep thrift | grep -v qlmaas)
tar xfz /var/qaptiva-slurm-installer/support/thrift/thrift-c++-0.20.0.tgz -C /usr
```

## Install the RPMS

For Qaptiva Slurm, the system hosting the Qaptiva Power Access Front-End services - the Appliance itself or the Power Access proxy system – is called the midtier.

The qlmaas312-slurm-midtier RPM must be installed on the midtier system.

Install the two other package groups slurm-loginnode and slurm-qlmproxy on the relevant systems as discussed above.

In case old RPMS are still installed, uninstall them.

```
dnf remove $(rpm -qa | grep -E "myqlm|qat|qlmaas|qaptiva" | grep -- -bull.1.9)
```

Then install the new RPMS.

```
# slurm-midtier
dnf install --enablerepo=qaptiva-slurm qlmaas312-slurm-midtier

# slurm-loginnode
dnf install --enablerepo=qaptiva-slurm qlmaas312-slurm-loginnode

# slurm-qlmproxy (compute node acting as a proxy to the Appliance)
dnf install --enablerepo=qaptiva-slurm qlmaas312-slurm-qlmproxy
```

Now that the software has been installed everywhere, it is time to configure Qaptiva Power Access to use the Slurm Scheduler.

## Configuration

Please refer to the document: Qaptiva_Power_Access_Admin_Guide-1.10.2.pdf for more details.

## /etc/qlmaas/django.py

This file is located on the system that manages Qaptiva Power Access front end services. Attention, python.py is a symbolic link. **Please do not break this link while editing the file**.

Replace or add the following sections. Make sure that you do not have multiple entries in the file (if there is, the last one in the file will override the previous ones).

```
SCHEDULER = "qat.qlmaas.slurm.scheduler:SLURM_SCHEDULER"
ADDITIONAL_SERVICES = {"QLMQueue": "qat.qlmaas.slurm.qlmqueue:QLM_QUEUE"}
SLURM_COMMANDS = {
    "HOST": "...",  # IP of the machine running slurm-commands (i.e. the login node)
    "PORT": 5250,   # default port
    "AUTH": {
        "CERTFILE": os.path.expanduser('~django/.ssl/certs/django.cer'),
        "KEYFILE": os.path.expanduser('~django/.ssl/private/django.key'),
        "CHECK_HOST": False
    }
}
SLURM_NODE_TO_WORKER = {
    "MyProxyQLM": "worker_0"
}
```

Note: SLURM_NODE_TO_WORKER is normally managed by the Qaptiva Installer.

- MyProxyQLM highlighted above is the name of the node name of the compute node that is labelled QAPTIVA; usualy this node name can be found in /etc/slurm/partitions.conf (see below).

- The "worker_id" on the right side must match the one under [qlm-worker] in /etc/qlmaas/qlmaas.ini in the Appliance (worker_0 is the default).

*Like its name implies, django.py is a python file (dictionary) before to be a configuration file. Consequently, python indentation must be respected.*

## /etc/qlmaas/client.ini

In the compute node to be used with Qaptiva Power Access also called qlmproxy, it is necessary to create /etc/qlmaas/client.ini.

```
cat /etc/qlmaas/client.ini
[Server]
hostname=<IP of Qaptiva Power Access Proxy>
check_host=False
authentication=ssl

[SSL]
certificate=~/.ssl/qlmaas.cer
key=~/.ssl/qlmaas.key
```

### /etc/slurm/partition.conf

This file – that must be identical on all system where the Slurm configuration resides - needs changes associated to the qlmproxy node (see definition in client.ini above).

• The qlmproxy node should have the **QAPTIVA** feature added to its list of features.

• The qlmproxy node must be configured with GRES corresponding to the number of CPUs, and the amount of memory on the corresponding Appliance.

• For remote exposing devices (GPUs or QPUs) on the Appliance, these devices must also be configured as GRES.

Here would be a node configuration example, exposing a Qaptiva Appliance with 192 CPUs, 6181471MB, 4 NVIDIA GPUs:

```
NodeName=MyProxyQLM CPUs=... Boards=1 SocketsPerBoard=8 CoresPerSocket=24 ThreadsPerCore=1 Real
Memory=... Features=QAPTIVA
Gres=qaptiva_memory_mb:6181471,qaptiva_cores:192,gpu-NVIDIA:4
```

The GRES field for GPUs or QPUs will be directly equivalent to devices available on the Appliance.

```
gpu-NVIDIA:4 means exposing 4 Device(type=DeviceType.GPU, manufacturer="NVIDIA")
```

### /etc/slurm/slurm.conf

Tell the cluster about all Gres resources that can potentially be declared in the partition.conf above.

```
GresTypes=qaptiva_memory_mb,qaptiva_cores[,gpu-NVIDIA],[qpu-EVIDEN]
```

### User certificate

To be able to submit jobs, a user must have a valid certificate created by qlmcerts on the Qaptiva Appliance.

```
qlmcerts -c <user>
```

This command automatically copies the certificate from under /etc/ssl/… to under ~user/.ssl.

### System certificate

Create the following certificate on the Qaptiva Power Access Proxy if not yet done; please note that the Qaptiva installer may have already created it depending on what were the selections at install time.

```
# List all certificates
qlmcerts --list
# If slurmcommands is not available, create it
qlmcerts --type server -c slurmcommands
```

When prompted for the IP address, put the address of the system where the qlmaas-slurmcommand service is running, i.e. The **Slurm login node**.

If the certificate above was not already installed in the login node by the Qaptiva Installer, install it now. This will allow the qlmaas-gunicorn service on the Qaptiva Power Access proxy and the qlmaas-slurmcommand service on the login node to talk with each other. On the login node, issue the following commands:

```
mkdir -p /etc/ssl/portal/certs/ /etc/ssl/qlm/services/private/

scp <qaptiva power access>:/etc/ssl/qlm/services/certs/services-chain.cer
/etc/ssl/portal/certs/django-services-chain.cer

scp <qaptiva power access>:/etc/ssl/qlm/services/certs/slurmcommands.cer
/etc/ssl/qlm/services/certs/slurmcommands.cer

scp <qaptiva power access>:/etc/ssl/qlm/services/private/slurmcommands.key
/etc/ssl/qlm/services/private/slurmcommands.key

cat /etc/ssl/qlm/services/certs/slurmcommands.cer >>/etc/ssl/portal/certs/django-services-chain.cer

systemctl restart qlmaas-slurmcommands
```

## Restart the services.

### Qaptiva Power Access Proxy

```
qlmsvc --restart
```

### Login node

```
systemctl restart qlmaas-slurmcommands [munge]
```

### Controller node

```
systemctl restart slurmctld [slurmdbd slurmd munge]
```

### Compute node

```
systemctl restart slurmd [munge]
```

### Testing

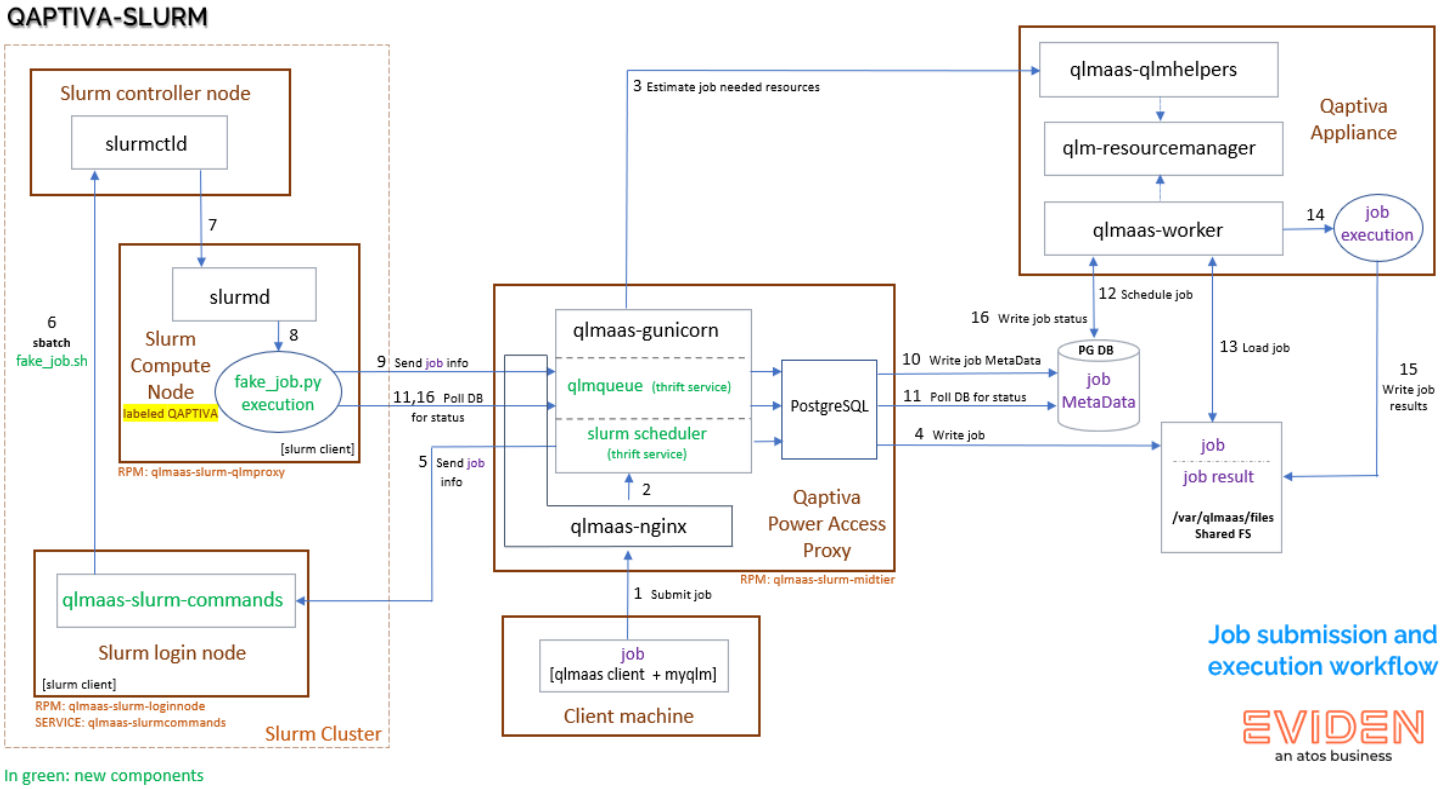
To evaluate that everything is working, run the runtests command from the Qaptiva Appliance. You can use qatuser, the Qaptiva pre-created test user, or create your own (if you create your own, see the discussion on user certificate above).

```
su - qatuser
runtests --qlmaas --host <qaptiva power access proxy IP>
```

- To trace activities and debug problems under Slurm, use slurm commands (sinfo, squeue, scontrol,…), systemd commands (systemctl, journalctl, …) and logfiles (/var/log/slurm, /tmp/job.out, …).

- To trace activities and debug problems under QLMaaS, use QLMaaS commands (qlminfo, qlmsvc, qlmcerts, …) , systemd commands (systemctl, journalctl, …) and logfiles (/var/log/qlmaas, /var/log/qlm-rm, /var/qlmlog/<user>,…).

# APPENDIX A




The boxes inside "Slurm cluster" in the diagram are normal Slurm nodes. We show three systems just because in a Slurm cluster, nodes can have different specialty functions, but everything can be in one node or a mixed of them.

The login node above is a system with just "Slurm core" installed (meaning no Slurm services are needed to run there). It will be used by the qlmaas-slurmcommand service to send jobs to slurmctld using the normal Slurm API commands (sinfo, sbatch, scontrol, …). **As such, it needs to be a part of the cluster, and be able to talk to the Gunicorn service on the Qaptiva Power Access proxy**.

Qaptiva Slurm needs one compute node per Qaptiva Appliance to be served. This compute node should be dedicated to Qaptiva only, i.e., Slurm should only use it to schedule quantum jobs. The resources on the Appliance (CPU, MEM, GPU and QPU) must be configured in the GRES, and the compute node feature must be set to 'QAPTIVA.' The resources on the compute node do not matter for Qaptiva. This node is also referred to as qlmproxy node (not to be confused with Qaptiva Power Access Proxy).

The Slurm "quantum" job is called "faked job"; despite for it to be a slurm job, it is not the real quantum job, but rather metadata for Qaptiva. Once the faked job is put in execution by Slurm, it tells Qaptiva Power Access to enter the job metadata in the default FIFO queue (PostgreSQL). Qaptiva is still the one polling for the real job to be executed (two-scheduler method). The Slurm quantum job (aka faked job) is told when the real quantum job is done, and Slurm can wrap up.

EVIDEN
an atos business

# APPENDIX B

## Qaptiva Slurm package groups with dependencies RPMS

**qaptiva3x-access (managed by Qaptiva Installer)**
- myqlm3x-comm
- myqlm3x-core
- qat-tutorial-doc
- qlmaas3x-client-light
- qlmaas3x-common
- qlmaas3x-server-common + qlmaas-server-common-config
- qlmaas-front
- qlmaas3x-server-django + qlmaas-server-django-config
- qlmaas3x-services
- qlmaas3x-thrift
- qlmaas-tutorial-notebooks
- qlmtools + qlmtools-cep

**qlmaas3x-slurm-midtier**
- qlmaas3x-slurm-qlmqueue
- qlmaas3x-slurm-commands

**qlmaas3x-slurm-loginnode** + qlmaas-slurm-loginnode-config
- myqlm3x-comm
- myqlm3x-core
- qlmaas3x-services
- qlmaas3x-slurm-commands
- qlmaas3x-slurm-common
- qlmaas3x-thrift

**qlmaas3x-slurm-qlmproxy**
- myqlm3x-comm
- myqlm3x-core
- qlmaas3x-client-light
- qlmaas3x-slurm-common
- qlmaas3x-slurm-qlmqueue
- qlmaas3x-thrift