

Cobalt Installation Instructions for Blue Gene/L Systems

(Updated for version 0.97.0)

- If upgrading from a version prior to 0.97.0pre1, refer to [this page](#) for preserving your system configuration.
- If upgrading from 0.94, run `misc/dumpoldqueue.py` to generate commands to rebuild the queue
Note that this command isn't installed by the rpm, and can be run directly out of the source distribution
- Install Prerequisites
 - ◆ Python 2.3+
 - ◆ PyOpenSSL
 - ◆ DB2-python

You may install Cobalt from RPMs or source.

RPMs

On SLES9, binary RPMS from the distribution site may be used. On the server side:

```
# rpm -ihv ftp://ftp.mcs.anl.gov/pub/cobalt/rpms/sles9-ppc64/PyOpenSSL-0.6-1.ppc64.rpm
# rpm -ihv ftp://ftp.mcs.anl.gov/pub/cobalt/rpms/sles9-ppc64/cobalt-0.97-1.ppc64.rpm
```

On the client and server side side:

```
# rpm -ihv ftp://ftp.mcs.anl.gov/pub/cobalt/rpms/sles9-ppc64/cobalt-clients-0.97-1.ppc64.rpm
```

Source

- Install prerequisites as described in the instructions for installing from an RPM
- Download the tarball from <ftp://ftp.mcs.anl.gov/pub/cobalt/> and extract it

Quick install:

```
# ./configure
# make
# make install
```

Custom install:

`./configure` has several options, see them all with '`./configure --help`'. Some interesting ones are:

- `--with-prefix=<path>` location for root of cobalt install `$prefix/bin`, `$prefix/sbin`, etc. (default `/usr/local`)

- `--with-wrapper=<path>` location for wrapper to be installed (default `$prefix/bin`)
- `--sysconfdir=<path>` location for configuration files, like `cobalt.conf`

There are two different make targets for installing just the clients or just the wrapper.

- `make install-clients`
- `make install-wrapper`

The default 'make install' installs all the daemons, admin tools, and user tools. Running 'make install-clients' will just install the user tools (cqstat, cqsub, etc.) and their man pages.

Running 'make install-wrapper' will install just the wrapper script, which is used to control access to the `cobalt.conf` file. See below for securing the wrapper and `cobalt.conf`.

Further Configuration

- Configure the component infrastructure
 - ◆ Generate an ssl key. Be sure to change the location strings to match your installation.

```
# openssl req -batch -x509 -nodes -subj "/C=US/ST=Illinois/L=Argonne/CN=$HOST" -days 1000
# openssl req -batch -new -subj "/C=US/ST=Illinois/L=Argonne/CN=$HOST" -key /etc/cobalt.k
```

- Copy the sample `cobalt.conf` file (`/etc/cobalt.conf.sample`) into place
- Set password in the `[communication]` section of this file to a secret, new value
- Change the service-location line in the `[components]` section to include the service node hostname
- **If upgrading from a previous version of cobalt, add the `[components]` section from the example `cobalt.conf` file with the URL for the service location facility set to the host where cobalt components run**
- `/etc/cobalt.conf` needs to be installed on all hosts that will run cobalt components or clients
- Create component persistence directory on the server side

```
# mkdir /var/spool/cobalt
```

- Create accounting log directory on the server side (defaults to `/var/log/cobalt-accounting`, otherwise the `log_dir` value in the `[cqm]` section of `cobalt.conf` is used)

```
# mkdir /var/log/cobalt-accounting
```

- Start the cobalt components

```
# /etc/init.d/cobalt start
```

This will start four components: `slp.py`, `bgpm.py`, `cqm.py`, and `bgsched.py`.

- Verify cobalt components Check that all components are functioning properly. The commands `slpstat.py`, `cqstat`, and `partadm.py -l` should all return no output and no errors.
slpstat was mistakenly not included in the 0.95.2 rpms. This will be fixed before 0.95.3

- Secure the cobalt component communication infrastructure Cobalt uses a shared secret key to authenticate clients. In order to protect this key from users, we install a setgid wrapper that calls all approved clients and can read the secret key. In order to set this up:

- ◆ Create a group for controlling access to /etc/cobalt.conf
- ◆ `chgrp newgroup /etc/cobalt.conf`
- ◆ `chgrp newgroup /prefix/bin/wrapper`
- ◆ `chmod 2755 /prefix/bin/wrapper`
- ◆ `chmod 640 /etc/cobalt.conf`

- Create partitions for the scheduler to use

- ◆ Create the partitions in the DB/2 database
- ◆ Register these partitions using `partadm`, and set dependencies appropriately

```
# /usr/sbin/partadm.py -a -s 32 a_1 a_2
# /usr/sbin/partadm.py -a -s 64 a_12
# /usr/sbin/partadm.py --deps=a_1:a_2 a_12
```

- ◆ Activate and enable these partitions. Active partitions are functional, but may not be scheduled. Enabled partitions should be scheduled by `bgsched`.

This example marks all partitions as functional (otherwise no jobs could run), but only enables the 64 node partition.

```
# /usr/sbin/partadm.py --activate a_1 a_2 a_12
# /usr/sbin/partadm.py --enable a_12
```

- If upgrading from cobalt-0.94, run the commands generated by `dumpoldqueue.py` now

At this point, jobs submitted to the queue manager should be detected and run by the scheduler. The scheduler uses the following characteristics to decide if a partition is suitable for a job.

- lack of any pertinent reservations
- membership in the job's queue
- partition size (jobs are slotted to the next partition size up from the size. ie 32 nodes jobs run on the 32 node partitions, 33 node jobs require a 64 node partition. 64 node jobs will not run on partitions larger than 64 nodes. This will need to be fixed to reflect the actual partition sizes available in the configuration.

Dynamic OS changes are not enabled by default, but support exists in `cqm`. If you want to run jobs with different kernels and ramdisks, three things must be configured:

- set the `bgkernel` option to true in `cobalt.conf`
- setup `/bgl/local/profiles` with one directory per boot profile
- create `/bgl/local/partitions` directory, and populate it with a link per partition. These links point at the current boot profile for a given partition
- set partitions to boot from these links in `mmcs`