SGE Cloud Plugin

Plugin Information
No information for the plugin 'sge-cloud-plugin' is available. It may have been removed from distribution.

This plugin submits builds to the Sun Grid Engine (SGE) batch scheduler. Both the open source version of SGE 2011.11 and the commercial Univa Grid Engine (UGE) 8.3.1 are supported.

> Distribution of this plugin has been suspended due to unresolved security vulnerabilities in the prerequisite Copy-To-Slave Plugin.

Features

This plugin adds a new type of build step *Run job on SGE* that submits batch jobs to SGE. The build step monitors the job status and periodically (default one minute) appends the progress to the build's *Console Output*. Should the build fail, errors and the exit status of the job also appear. If the job is terminated in Jenkins, it is also terminated in SGE.

Builds are submitted to SGE by a new type of cloud, *SGE Cloud*. The cloud is given a label like any other agent. When a job with a matching label is run, *SGE Cloud* submits the build to SGE.

Files can be uploaded and sent to SGE before the execution of the job and downloaded from SGE after the job finishes. Currently this feature only supports shared file systems.

An email can optionally be sent when the job finishes.

Configure SGE

In SGE, add your Jenkins master host as an SGE submit host.

Configure Jenkins

Set Environment Variables

SGE requires some environment variables. There are various other ways to add Jenkins environment variables, but the following method is one of the most dependable.

In *Manage Jenkins > Configure System*, add *Environment Variables*:

For open source SGE:

Name	Value
`SGE_ROOT`	`/path/to/sge`
`SGE_BIN`	`/path/to/sge/bin/linux-x64`

For commercial UGE:

Name	Value
`SGE_ROOT`	`/path/to/uge`
`SGE_BIN`	`/path/to/uge/bin/lx-amd64` or `.../darwin-x64` or `.../win-x86`
`SGE_CELL`	Your cell name
`SGE_CLUSTER_NAME`	Your cluster name
`SGE_EXECD_PORT`	`64455`
`SGE_QMASTER_PORT`	`64444`

Environment variable troubleshooting

The SGE error message:

```
Unable to initialize environment because of error: cell directory "/path
/to/sge/default" doesn't exist
```

means that `SGE_CELL` is undefined (it defaults to `default`).

Create the cloud

In *Manage Jenkins > Configure System*, add a new cloud of type *SGE Cloud*. Fill in the required information for the newly created cloud.

Set up a project to run on SGE

In a project, specify the *Label* that you specified in *SGE Cloud*.

Add a *Run job on SGE* build step and specify the batch job script you want to run on SGE.

Now, when Jenkins runs the project, it will run on the *SGE Cloud* with the matching label.

Set your script to fail on the first failure

By default, the exit status of the last command determines the success or failure of the build step. For example, the following script would be inappropriately considered a success:

```
ls /nonexistent    # Error, exit status 2
echo "This echo command succeeds with exit status 0 despite the error
on the previous line"
```

If you prefer that your job fail and halt upon the first nonzero exit status, use the Bash -e option. The following script will fail upon the first error:

```
set -e
ls /nonexistent    # Error, exit status 2
echo "This is never executed because the above ls command failed."
```

Additional qsub options

So that you can see the `qsub` command that was used to submit jobs, the SGE Plugin prints the qsub command to the Jenkins job *Console Output*.

```
Submitting SGE job using the command:
    "$SGE_BIN/qsub" ...    # Options not shown in docs because they
will undoubtedly be out-of-date
```

It is possible to specify additional `qsub` command line options within the *Run job on SGE* build script on lines beginning with #$. For example:

```
#$ -P project_name
```

While this might sometimes be useful, it can cause trouble if your *Run job on SGE* build step **inadvertently** contains #$. In particular, this can happen if you comment out a line that begins with $:

```
#$SOME_COMMAND
```

There is no such `qsub` command line option `SOME_COMMAND`, so you get the unhelpful message:

```
qsub: Unknown option
```

Job States

Unfinished jobs

The qstat man page describes the following job states (job status) defined in SGE. Each state is a string whose first character is most meaningful:

- "d", for deletion
- "E", for error
- "h", for hold
- "r", for running
- "R", for restarted
- "s", for suspended
- "S", for queue suspended
- "t", for transfering
- "T", for threshold
- "w", for waiting

Naturally, `qstat` can only describe jobs that were actually submitted to SGE. The SGE Cloud Plugin defines an additional state for jobs it could not even submit to SGE:

- "J", for Jenkins SGE plugin failure to submit the job

Finished jobs

SGE `qstat` states cover only unfinished jobs, yet the SGE Cloud Plugin expects that finished jobs also have a state. Therefore the SGE Cloud Plugin uses the shell exit status as the state of the finished job:

- "0" (zero) for a successfully finished job
- "1" through "255" for a job that failed with a nonzero exit status

Exit status above 128 indicates that a signal terminated the job. See Job Exit Status for an explanation of some exit statuses.

Viewing the Job Workspace

Each project has a *Workspace* button that you can use to view the project workspace files in your web browser. This handy feature relies on the slave that executed the job. SGE slaves are reused and if kept busy they can live a long and productive life. However, slaves left idle for an extended time are deleted. Once the slave is gone, the *Workspace* button will no longer work. Then the files can only be viewed using other methods like the command line.

In J*enkins > Manage Jenkins > Configure System > SGE Cloud*, the *Maximum idle time* field controls how long idle slaves are retained. If you find that slaves disappear while you still want to view the workspace, increase *Maximum idle time*.

Environment Variables

Jenkins adds environment variables to the environment, and these are imported into the SGE job environment. Then SGE adds some more. There is just one variable name collision: `JOB_NAME`. So before SGE overwrites Jenkins' `JOB_NAME`, the Jenkins value is preserved in environment variable `JENKINS_JOB_NAME`.

Project History

LSF Cloud Plugin

`sge-cloud-plugin` was forked from lsf-cloud-plugin and modified to work with SGE instead of LSF.

SGE Cloud Plugin

`sge-cloud-plugin` is being used in industrial production at Wave Computing.

While it might be nice to integrate `sge-cloud-plugin` and `lsf-cloud-plugin` into a single Jenkins plugin, this would be difficult to test, as few organizations have all batch systems installed.

Condor Cloud Plugin (future)

From time to time people inquire about a Condor version of this plugin. To create this you would fork the SGE plugin, then replace the SGE commands it sends with Condor commands. No official Jenkins Condor Plugin has materialized, but potential candidates do turn up in a search of GitHub. Good luck.