IBM Spectrum LSF
Version 10 Release 1.0

Using the IBM Spectrum LSF Resource Connector
IBM Spectrum LSF
Version 10 Release 1.0

Using the IBM Spectrum LSF Resource Connector

IBM
Note
Before using this information and the product it supports, read the information in "Notices" on page 95.

This edition applies to version 10, release 1 of IBM Spectrum LSF (product numbers 5725G82 and 5725L25) and to all subsequent releases and modifications until otherwise indicated in new editions.

Significant changes or additions to the text and illustrations are indicated by a vertical line (|) to the left of the change.

If you find an error in any IBM Spectrum Computing documentation, or you have a suggestion for improving it, let us know.

Log in to IBM Knowledge Center with your IBMid, and add your comments and feedback to any topic.

© Copyright IBM Corporation 2015, 2017.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
# Contents

**Chapter 1. Resource connector for IBM Spectrum LSF overview** .......................... 1

**Chapter 2. Updating LSF configuration for resource connector** ......................... 5
- Configure resource providers ........................................... 8
- Pre-provisioning and post-provisioning ................................ 9
- Configure resource provisioning policies .............................. 11
- Use the LSF patch installer to update resource connector ........ 12

**Chapter 3. Configuring IBM Spectrum Conductor with Spark for LSF resource connector** .................................................. 15
- Managing resource sharing and distribution .......................... 15
- Installing LSF on IBM Spectrum Conductor with Spark compute hosts .................................................. 17
- Configuring LSF resource connector for IBM Spectrum Conductor with Spark ............................ 18

**Chapter 4. Configuring OpenStack for LSF resource connector** ....................... 21
- Configuring the DNS server for OpenStack .......................... 23
- Configuring LSF resource connector for OpenStack ............... 24

**Chapter 5. Configuring Amazon Web Services for LSF resource connector** .......... 27
- Configuring AWS access with federated accounts .................. 32
- Configuring LSF resource connector for Amazon Web Services .................................................. 33
- Configuring user scripts to register Amazon Web Services hosts with the LSF master host .................. 34
- Use AWS Spot instances .................................................. 35

**Chapter 6. Configuring Microsoft Azure for LSF resource connector** ............... 39
- Configuring LSF resource connector for Microsoft Azure ........ 42

**Chapter 7. Configuring IBM Bluemix for LSF resource connector** .................. 45
- Configuring LSF resource connector for IBM Bluemix .......... 48

**Chapter 8. Submit jobs to LSF resource connector** ........................................ 51
- Submitting jobs to borrow resources through EGO ............... 51
- Submitting jobs to launch instances from OpenStack .......... 53
- Submitting jobs to launch instances from Amazon Web Services .................................................. 54
- Submitting jobs to launch instances from Microsoft Azure .................................................. 56
- Submitting jobs to launch instances from IBM Bluemix ........ 57

**Chapter 9. Logging and troubleshooting the LSF resource connector** ................. 59

**Chapter 10. Reference for LSF resource connector** .......................................... 61
- lsb.applications .................................................. 61
- lsb.queues .................................................. 61
- lsf.conf .................................................. 63
- egoprov_config.json ............................................. 66
- egoprov_ego.conf ................................................ 68
- egoprov_templates.json ............................................ 69
- hostProviders.json ................................................ 71
- osprov_config.json ................................................ 73
- osprov_templates.json ............................................. 75
- policy_config.json ................................................ 77
- awsprov_config.json ............................................. 79
- awsprov_templates.json ............................................ 81
- azureprov_config.json ............................................ 86
- azureprov_templates.json ............................................ 87
- softlayerprov_config.json .......................................... 90
- softlayer_templates.json ............................................ 91

**Notices** .................................................. 95
- Trademarks .................................................. 97
- Terms and conditions for product documentation .............. 97
- Privacy policy considerations ....................................... 98

© Copyright IBM Corp. 2015, 2017
Chapter 1. Resource connector for IBM Spectrum LSF overview

The resource connector for IBM Spectrum LSF feature (also referred to as *host factory*) enables LSF clusters to borrow resources from supported resource providers.

LSF resource connector plug-ins supports the following resource providers:
- IBM Spectrum Conductor with Spark and IBM Spectrum Symphony through EGO, configured with the egoprov_config.json and egoprov_templates.json files.
- OpenStack, configured with the osprov_config.json and osprov_templates.json files.
- Amazon Web Services (AWS), configured with the awsprov_config.json and awsprov_templates.json files.
- IBM Bluemix (formerly SoftLayer), configured with the softlayerprov_config.json and softlayerprov_templates.json files.
- Microsoft Azure, configured with the azureprov_config.json and azureprov_templates.json files.

The `LSB_RC_EXTERNAL_HOST_FLAG` parameter in the lsf.conf enables the resource connector feature. The value of the parameter must be set to a Boolean resource name in the lsf.shared file to identify borrowed hosts, for example, `LSB_RC_EXTERNAL_HOST_FLAG=conductorhost` for borrowing through EGO.

You must make sure that the module `schmod_demand` is enabled in the PluginModule section of the lsb.modules file (it is commented out in the file at installation by default).

LSF clusters can borrow hosts from a resource provider to satisfy pending workload. The borrowed resources join the LSF cluster as hosts. When the resources become idle, LSF resource connector returns them to the resource provider.

The resource connector generates requests for extra hosts from the resource provider and dispatches jobs to dynamic hosts that join the LSF cluster. When the resource provider reclaims the hosts, the resource connector requeues the jobs that are running on the LSF hosts, shuts down LSF daemons, and releases the hosts to the provider.

**How LSF borrows hosts from a resource provider**

The following workflow summarizes how your job uses resources that are borrowed from a resource provider:

1. A user submits a job to LSF as normal. The job generates demand, but the cluster doesn’t have enough resources to service it, so it must borrow hosts from an external provider.
2. The `mbatchd` daemon checks if hosts are already allocated that match the demand, and LSF calculates how many of each template type it requires to run the job. LSF sends this demand to the resource connector `ebrokerd` daemon.
The administrator configures templates representing LSF hosts. Each resource provider has its own template file that defines the mapping between LSF resource demand requests and hosts that the provider allocates to LSF. Each template in the file represents a set of hosts that share some attributes, such as the number of CPUs, the amount of available memory, the installed software stack, and operating system image.

3. Based on the demand from the submitted job, LSF resource connector makes an allocation request to the resource provider. For example, if the resource provider is EGO, resource connector makes an allocation request to EGO as the LSF_Consumer.

4. For EGO resources, if enough resources are available in the rg_shared resource group, the allocation request succeeds.

5. The ebrokerd daemon monitors the status of the request in the resource provider until it detects that the request succeeds, starts LSF daemons on the allocated hosts, and notifies LSF that the hosts join the cluster and are ready to use.

6. When the host joins the cluster, the job is dispatched to the host.

7. When there is no more demand for borrowed resources, LSF lets resource connector know and the ebrokerd daemon returns the resources to the provider.

8. Some resource providers (for example, EGO) can also be configured to reclaim borrowed resources from LSF when they require the resources to satisfy their workload demand.

**Example of borrowing hosts from EGO**

In the following example, the resource provider is EGO:

1.  
   ```
   bhosts -a
   HOST_NAME     STATUS  JL/U  MAX NJOBS  RUN SSUSP USUSP RSV
   lsfmaster     ok      -     1   1    0   0   0
   
   ego01         ok      -     1   1    0   0   0
   ```

2. EGO has a host ego01 with ncpus=1.

3. Resource connector is configured to connect to the EGO cluster.

4. A template is created that provides a numeric attribute ncpus with range [1:1].

5. The bsub command submits a job that requires a single slot.

6. Eventually host ego01 joins the cluster, and the new job runs.

   ```
   bhosts -a
   HOST_NAME     STATUS  JL/U  MAX NJOBS  RUN SSUSP USUSP RSV
   lsfmaster     ok      -     1   1    0   0   0
   ego01         ok      -     1   1    0   0   0
   ```

**Limitations**

Do not create advanced reservations on AWS instances because the reservations might be terminated after idle time. If advanced reservations are created on instances, they remain active if the instances are destroyed. However, jobs are not able to run on the instance since the LSF daemons are shut down on terminated instances and the jobs become unavailable. Hosts can be returned to their resource provider at any time by idle time or time-to-live policy, EGO reclaim, or AWS reclaim. The hosts might be closed or unavailable when the advanced reservation starts.

It also is possible for resource connector to over-demand for its workload if an OpenStack VM joins the cluster but is not immediately usable by scheduler.
If borrowed hosts cannot resolve each other's host name, then commands like `lsrerp` do not work when used to copy the files from one instance to another.

The `HOSTS` parameter in the `lsb.queues` file and the job-level `-m` option do not apply to borrowed hosts managed through the resource connector.

Administrators must use the `RC_HOSTS` parameter in the queue to specify the external resources that resource connector can borrow resources from. A queue can borrow hosts only from the resource that the `RC_HOSTS` parameter defines. For example, if the queue defines only the AWS resources (`RC_HOSTS=awshost`), it cannot borrow EGO or OpenStack resources.

The `RC_ACCOUNT` parameter that is defined in an application profile in the `lsb.applications` file is not displayed in the `bapp -l` command. The `bqueues -l` command shows the value of the `RC_ACCOUNT` and `RC_HOSTS` parameters that are defined in queues.
Chapter 2. Updating LSF configuration for resource connector

Configure LSF to enable the resource connector.

Fast path:

For AWS resources only, you can use the aws_enable.sh script and the aws_enable.config file to automate AWS setup for steps 3 to 8. After installation, these files are located in <LSF_TOP>/lsf_version/install. The steps for using the aws_enable.sh script are described in the aws_enable.config file.

The aws_enable.sh script updates the following files:
- <LSF_TOP>/conf/resource_connector/aws/conf/awsprov_config.json
- <LSF_TOP>/conf/resource_connector/aws/conf/credentials
- <LSF_TOP>/<LSF_VERSION>/resource_connector/aws/scripts/user_data.sh

These files are backed up to <file_name>.aws_backup before they are updated by the aws_enable.sh script. All updated files are rolled back if script fails.

To use the aws_enable.sh script, follow these steps:
1. Source the LSF profile (for example, <LSF_TOP>/conf/profile.lsf).
2. Edit the <LSF_TOP>/10.1/install/aws_enable.config file to configure the installation parameters.
3. Run ./aws_enable.sh -f aws_enable.config.

If live configuration in LSF is enabled, the lsb.queues and lsf.cluster.cluster_name under the LSF_LIVE_CONFDIR directory is updated by the aws_enable.sh script.

To disable the LSF resource connector for AWS after you run the aws_enable.sh script, comment out or remove the line in the lsf.conf where the AWS host resource is defined (in the LSB_RC_EXTERNAL_HOST_FLAG=awshost parameter).

Step 9 is recommended, and step 10 is optional. Restart the LSF daemons on the master host for the changes to take effect.
1. Log in to the LSF master host as root.
2. Configure DNS.
   LSF looks up host names and addresses for all communication between hosts borrowed from a resource provider and master host candidates. Make sure that your environment settings for IP address resolution work between the LSF master host and borrowed hosts that join the cluster.
   Check whether the master host candidates can ping the borrowed hosts by using its host name from the hostname command and vice versa. Make sure that the borrowed hosts can ping each other with both public IP address and reported host name. If the hosts can ping each other only by using IP address but not by using the host name, DNS settings need to be configured.
   If the borrowed host cannot join the cluster, check the VPN, firewall, or security group settings. You must open firewall rules for all LSF ports between the LSF master and borrowed slave host IP ranges.
3. Make sure that the resource connector demand calculation scheduler module is configured.
Fast path: If you use the aws_enable.sh script and the aws_enable.config file to automate AWS setup, the schmod_demand plug in is uncommented in the lsb.modules file.

Check the lsb.modules file. By default, the module schmod_demand is not enabled (it is commented out in the file at installation). To enable resource connector, make sure that the schmod_demand is configured in the file. If not, add it to the PluginModule section.

```
Begin PluginModule
  SCH_PLUGIN  RB_PLUGIN  SCH_DISABLE_PHASES
  ...
  schmod_demand () ()
End PluginModule
```

4. Define Boolean resources to identify hosts that are borrowed from resource providers.

Fast path: If you use the aws_enable.sh script and the aws_enable.config file to automate AWS setup, the awshost resource is added to the lsf.shared file.

Add new resources to the LSF resource list so that they are accepted by LSF. On the LSF master host, modify the lsf.shared file to add Boolean resources to the Resource section. The following example defines the conductorhost, oshost, and awshost resource names.

```
Begin Resource
  RESOURCENAME  TYPE  INTERVAL  INCREASING  DESCRIPTION
  ...
  conductorhost  Boolean  ()  ()  (Hosts borrowed from EGO)
  oshost        Boolean  ()  ()  (Hosts borrowed from OpenStack)
  awshost       Boolean  ()  ()  (Hosts borrowed from AWS)
  azurehost     Boolean  ()  ()  (instances borrowed from Azure)
  softlayercomp Boolean  ()  ()  (Soft_layer compute host)
  ...
End Resource
```

5. Enable the resource connector.

Fast path: If you use the aws_enable.sh script and the aws_enable.config file to automate AWS setup, the LSB_RC_EXTERNAL_HOST_FLAG=awshost parameter is configured in the lsf.conf file.

Define the LSB_RC_EXTERNAL_HOST_FLAG parameter in the lsf.conf file to enable the resource connector feature. The value of the parameter must be set to the Boolean resource name that you created to identify borrowed hosts, for example, LSB_RC_EXTERNAL_HOST_FLAG=conductorhost,

```
  LSB_RC_EXTERNAL_HOST_FLAG=oshost, LSB_RC_EXTERNAL_HOST_FLAG=awshost,
  LSB_RC_EXTERNAL_HOST_FLAG=azurehost, or
  LSB_RC_EXTERNAL_HOST_FLAG=softlayercomp.
```

6. Define the RC_HOSTS parameter in the lsb.queues for a queue for which borrowing needs to be enabled.

Fast path: If you use the aws_enable.sh script and the aws_enable.config file to automate AWS setup, an AWS queue awsexample is created with the RC_HOSTS=awshost parameter configured in the lsb.queues file.

the RC_HOSTS parameter also enables the queue to use the hosts that are already borrowed into the cluster.

```
RC_HOSTS = none | all | host_type [host_type ...]
```

The host_type flag is a Boolean resource that is a member of the list of host resources that are defined in the LSB_RC_EXTERNAL_HOST_FLAG parameter in the lsf.conf file.
If the `RC_HOSTS` parameter is not defined in the queue, its default value is `none`. Borrowing is disabled for any queue that explicitly defines `RC_HOSTS=none`, even if the `LSB_RC_EXTERNAL_HOST_FLAG` parameter is defined in the `lsf.conf` file.

If the `RC_HOSTS` parameter is not defined in any queue, borrowing from resource providers is disabled for all jobs.

7. If LSF was not installed with the dynamic host feature enabled (that is, the `ENABLE_DYNAMIC_HOSTS` parameter was not set in the `install.config` to enable dynamic hosts at installation), set the `LSF_DYNAMIC_HOST_WAIT_TIME` parameter in the `lsf.conf` file.

**Fast path:** If you use the `aws_enable.sh` script and the `aws_enable.config` file to automate AWS setup, the `ENABLE_DYNAMIC_HOSTS=Y` and `LSF_REG_FLOAT_HOSTS=Y` parameters are set in the `lsf.conf` file. The `LSF_DYNAMIC_HOST_WAIT_TIME=2` parameter is also configured in the `lsf.conf` file if the parameter is not already set. If the the `LSF_DYNAMIC_HOST_WAIT_TIME` parameter is already set, the script keeps the configured value.

To ensure the smooth operation of the borrowed hosts, set the `LSF_DYNAMIC_HOST_WAIT_TIME` parameter to a small value, such as 1 or 2 seconds.

8. Define the `LSF_HOST_ADDR_RANGE` parameter in the `lsf.cluster.cluster_name` file to enable security.

**Fast path:** If you use the `aws_enable.sh` script and the `aws_enable.config` file to automate AWS setup, the `LSF_HOST_ADDR_RANGE` parameter is configured in the `lsf.cluster.cluster_name` file to the value that you set in the `aws_enable.config` file.

If you do not set a value in the `aws_enable.config`, the script does nothing for the `LSF_HOST_ADDR_RANGE` parameter.

Only hosts with IP addresses within the specified range can be added to or removed from the AWS instance.

In the following example, all instances belonging to domains starting with `12.23.45` are allowed.

```
Begin Parameters
LSF_HOST_ADDR_RANGE=12.23.45.*  # ipaddress range of the AWS instance
End Parameters
```

9. Define the `RC_DEMAND_POLICY` parameter in the `lsb.queues` file to specify threshold conditions for triggering demand to borrow resources through resource connector for all the jobs in a queue.

The `RC_DEMAND_POLICY` parameter has the following syntax:

```
RC_DEMAND_POLICY = THRESHOLD[ [ num_pend_jobs, duration ] ... ]
```

The demand policy defined by the `RC_DEMAND_POLICY` parameter can contain multiple conditions, in an OR relationship. A condition is defined as `[ num_pend_jobs, duration ]`. The queue has more than the specified number of eligible pending jobs that are expected to run at least the specified duration in minutes. The `num_pend_jobs` option is required, and the duration is optional.

The default threshold is `THRESHOLD[[ 1,0 ]]`.

In the following example, LSF calculates demand if the queue has 5 or more pending jobs in past 10 minutes, or 1 or more pending jobs in past 60 minutes, or 100 or more pending jobs.

```
RC_DEMAND_POLICY = THRESHOLD[ [ 5, 10 ] [ 1, 60 ] [100] ]
```
As long as pending jobs at the queue meet at least one threshold condition, LSF expresses the demand to resource connector to trigger borrowing.

10. Optional: Define the RC_ACCOUNT parameter in the lsb.queues or lsb.applications file to tag the borrowed hosts so they cannot be used by other groups, users, or jobs.

**Important:** The RC_ACCOUNT parameter is supported only for the OpenStack and AWS resource providers. It is not supported for IBM Spectrum Conductor with Spark.

When a job is submitted to the queue or application, the host that the job borrows is tagged with the value of the RC_ACCOUNT. Other applications or queues that have a different value for the RC_ACCOUNT parameter cannot use the borrowed host.

When the borrowed host joins the cluster, you can use the `lshosts -s` or `lshosts -l` command to view its RC_ACCOUNT value.

For AWS and OpenStack, edit the user_data.sh file to use the `lshosts` command to see the RC_ACCOUNT value.

The user_data.sh script is located in the `<LSF_TOP>/LSF_VERSION/resource_connector/aws/scripts` directory.

a. Make sure that the LSF_TOP variable points to the LSF_TOP directory for your cluster.

b. Make sure that the following lines are not commented out.

```
%EXPORT_USER_DATA%

logfile=/tmp/userscript.log
env > $logfile
if [ -n "${rc_account}" ]; then
    sed -i "s/\(LSF_LOCAL_RESOURCES=.*\)/\1 [resourcemap ${rc_account}*rc_account]\"/" $LSF_CONF_FILE
    echo "update LSF_LOCAL_RESOURCES lsf.conf successfully, add [resourcemap ${rc_account}*rc_account]" >> $logfile
fi
```

Restart the LSF daemons on the master host for the changes to take effect.

```
lsadmin limrestart
lsadmin resrestart
badmin mbdrestart
```

---

**Configure resource providers**

You can configure multiple resource providers in one LSF cluster.

Each provider has its own set of configurable parameters that are defined in the hostProviders.json file. A log file for each provider (`<provider_name>-provider.log.<host_name>`) is located in the LOGDIR and a persistence file is located in LSF_SHAREDIR/<<cluster_name>>/resource_connector/<<providerName>>-db.json.

All the providers must have the same LSF administrator account.

Each host provider must have a unique name. If two different host providers use the same name, LSF logs a warning and ignores one of the entries.

The LSF administrator must have access to the directories specified by the confPath and scriptPath attributes in the hostProviders.json file.
You can specify an absolute path for configuration and script files. The default assumes a path relative \texttt{LSF\_TOP}/\texttt{conf/resource\_connector} for configuration files and \texttt{LSF\_TOP/LSF\_VERSION/resource\_connector/} for scripts.

\textbf{Related reference:}

“hostProviders.json” on page 71

\section*{Pre-provisioning and post-provisioning}

Set up pre-provisioning in LSF resource connector to run commands before the resource instance joins the cluster. Configure post-provisioning scripts to run cleanup commands after the instance is terminated, but before the host is removed from the cluster.

The pre- and post-provisioning script knows the instance ID, the instance name, and the instance IP address.
The pre-provisioning script runs on the master host right after the resource instance is created, but before the instance is marked as allocated to the LSF cluster, and before a job can start to run on it. Use the pre-provisioning script to run instance setup scripts (for example, network or user access configuration), run data transfer commands before the job starts on the instance, or add hosts to a specific group.

The post-provisioning script runs after the instance is terminated, but before it is relinquished to the provider. Use the post-provisioning script to run clean up tasks; for example, clean up job files or remove the host from the host cache file when the instance terminates.

You can specify a delayOnReturn attribute in your scripts that specifies the number of minutes that the resource connector waits before it returns the host in case the pre- or post-provisioning script fails. The default value is 20. For the scripts that are run successfully, resource connector does not apply the delay, even if delayOnReturn is set.

Enable pre- and post-provisioning

To enable pre- and post-provisioning scripts, set the following parameters in the hostProviders.json file:

preProvPath
Resource connector runs the pre-provisioning script that is specified with absolute path after the instance is created and started successfully but before it is marked allocated to the LSF cluster.

postProvPath
Resource connector runs the post-provisioning script that is specified with absolute path after the instance is terminated successfully but before it is removed from the LSF cluster.

provTimeOut
This parameter is used to avoid the pre- or post-provisioning program from running for unlimited time. Specify a value in minutes.

If the program doesn’t complete in the specified time, it is ended and reported as failed. You can disable pre- or post-provisioning by setting the provTimeOut value to 0.

The default value is 10 minutes. If the pre- or post-provisioning program doesn’t return after 10 minutes, it ends.

Pre-provisioning and post-provisioning script output

Pre and post provision scripts are expected to return results that look like the following example:

```json
[]
"machines": [
  {
    "name": "instance name",
    "result": "succeed",
    "message": "User added successfully"
  },
  {
    "name": "ip-192-168-0-154.us-west-2.compute.internal",
    "result": "failed",
    "message": "Could not resolve host name",
  }
]
```
"delayOnReturn" : 30
]
}

Example input.json file

The input.json file contains the following information:

```
[
  "machines": [
    {
      "name": "ip-192-168-0-153.us-west-2.compute.internal",
      "publicIpAddress": "55.66.xx.xx",
      "privateIpAddress": "55.66.xx.xx",
      "machineId": "i-19034xxxxx",
      "rcAccount": "project1",
      "providerName": "aws",
      "providerType": "awsProv",
      "templateName": "Template-VM-2"
    }
  ]
]
```

Example

```
#!/bin/sh
inputFile=$1
outputFile=$2
echo $inputFile $outputFile
count=$(cat $inputFile | jq '.machines[]' | grep 'name' | wc -l)
a=0
result="succeed"
while [ $a -lt $count ]
do
  hostName=$(cat $inputFile | jq '.machines[$a].name')
  privateIp=$(cat $inputFile | jq '.machines[$a].publicIpAddress')
  publicIp=$(cat $inputFile | jq '.machines[$a].privateIpAddress')
  instanceId=$(cat $inputFile | jq '.machines[$a].machineId')
  rcAccount=$(cat $inputFile | jq '.machines[$a].rcAccount')
  providerName=$(cat $inputFile | jq '.machines[$a].providerName')
  providerType=$(cat $inputFile | jq '.machines[$a].providerType')
  templateName=$(cat $inputFile | jq '.machines[$a].templateName')

  #add your custom code here for each machine in the request
  #write the output of each machine to the output json file
  sed -i '/}/i {"name": "$hostName", "result": "$result", "message": "$message"}
  a=`expr $a + 1`
done
```

Related reference:

"hostProviders.json" on page 71

Configure resource provisioning policies

LSF resource connector provides built-in policies for limiting the number of instances to be launched and the maximum number of instances to be created. The default plugin framework is a single python script that communicates via stdin and stdout in JSON data structures. LSF resource connector provides an interface for administrators to write their own resource policy plugin.
**Built-in policies**

LSF resource connector provides the following built-in policies, which you can configure in the policy_config.json file:

- Step index and step time determine how many instances to launch at a time.
  The **StepValue** parameter specifies step index and step time. For example, the "StepValue": "10:10" means that the resource connector launches no more than 10 instances every 10 minutes.
- Maximum number of instances (per account, per template, per provider) at any given time is specified by the **MaxNumber** parameter.

To enable the feature, you create a policy_config.json file under the LSF_TOP/conf/resource_connector directory and set the cluster administrator as the file owner.

**Resource policy plug in interface**

You should not change the default resource policy plugin files (Main.py and Log.py in the LSF_TOP/LSF_VERSION/resource_connector/policy directory). Instead, you can create your own script or binary executable file and specify the path of that script in the UserDefinedScriptPath parameter in the policy_config.json. The default policy script provided by LSF runs your script and uses the demand calculated by your script to create hosts for the cluster.

Your script can have the following input for each provider, template and **RC_ACCOUNT**:

- The demand target requested by the cluster.
- The current allocation of hosts.
- The outstanding requests that have been made but not yet filled.
- The number of reclaimed hosts.

**Related reference:**

"policy_config.json" on page 77

---

**Use the LSF patch installer to update resource connector**

**Update resource connector**

The LSF patch installer (patchinstall command) doesn’t support installing files under the LSF_TOP/conf directory. The patchinstall command operates only on files under the LSF_TOP/<version> directory.

When the patch installer installs new configuration files or other items under LSF_TOP/10.1/resource_connector/<provider_name>/conf, you must move them to the appropriate directory under LSF_TOP/conf/resource_connector/ <provider_name>/conf. You must also change the ownership of any new files and directories to the cluster administrator. Everything under the LSF_TOP/conf/resource_connector directory must be owned by the cluster administrator.

For example, when the patch installer installs the following new configuration files for Amazon Web Services (AWS) under LSF_TOP/10.1/resource_connector/aws/conf/:

- awsprov_templates.json
- awsprov_config.json
• credentials

You must move these files to the LSF_TOP/conf/resource_connector/aws/conf directory and change the ownership of the aws directory and configuration files to the cluster administrator.

**Roll back a resource connector patch**

Follow these steps to roll back a patch:

1. Log on to the LSF master host as root
2. Set your environment:
   - For **csh** or **tcsh**: `% source LSF_TOP/conf/cshrc.lsf`
   - For **sh**, **ksh**, or **bash**: `$ . LSF_TOP/conf/profile.lsf`
3. Close all hosts and queues on your cluster:
   - `badmin hclose all`
   - `badmin qinact all`
4. Roll back the patch:
   - `./patchinstall -r <patch>`
5. Shut down the cluster:
   - `badmin hshutdown all`
   - `lsadmin resshutdown all`
   - `lsadmin limshutdown all`
6. Restart the cluster:
   - `lsadmin limstartup all`
   - `lsadmin resstartup all`
   - `badmin hstartup all`
7. Open hosts and queues again:
   - `badmin hopen all`
   - `badmin qact all`

**Remember**: Remove the most recent patch and return the cluster to the previous patch level. To roll back multiple versions, you must roll back one patch level at a time, in the reverse order of installation.

To roll back the same version of the patch applied on multiple platforms, you must roll back the same patch for multiple packages you applied, in the reverse order of installation, so that the resource connector common files are also rolled back to previous version.

For more information about patch rollback, see the `-r` option of the `patchinstall` command in the *LSF Command Reference.*
Chapter 3. Configuring IBM Spectrum Conductor with Spark for LSF resource connector

Follow these steps to configure new resource groups and instance groups in your IBM Spectrum Conductor with Spark cluster so that the resource connector to make allocation requests on behalf of LSF.

If you do not have an existing IBM Spectrum Conductor with Spark cluster, follow the steps in the IBM Spectrum Conductor with Spark installation guide to install IBM Spectrum Conductor with Spark: [www.ibm.com/support/knowledgecenter/SS4H63_2.1.0/install/install.html](http://www.ibm.com/support/knowledgecenter/SS4H63_2.1.0/install/install.html)

Managing resource sharing and distribution

Configure new resource groups and instance groups in your IBM Spectrum Conductor with Spark cluster so that the resource connector to make allocation requests on behalf of LSF.

If you do not have an existing IBM Spectrum Conductor with Spark cluster, follow the steps in the IBM Spectrum Conductor with Spark installation guide to install IBM Spectrum Conductor with Spark: [http://www.ibm.com/support/knowledgecenter/SSVH2B_1.1.0/install/install.dita](http://www.ibm.com/support/knowledgecenter/SSVH2B_1.1.0/install/install.dita)

To have LSF borrow hosts from shared resource groups, create new LSF consumers and change the corresponding resource plan to share resources with the new LSF consumers.

1. Create a shared resource group.
   a. Create at least one resource group to share with LSF.
   b. Add one or more shared compute hosts to this resource group.
      For more information about creating resource groups, see [http://www.ibm.com/support/knowledgecenter/SSVH2B_1.1.0/manage_resources/resource_groups_create.dita](http://www.ibm.com/support/knowledgecenter/SSVH2B_1.1.0/manage_resources/resource_groups_create.dita)
      For example, create a new resource group `rg_shared` in IBM Spectrum Conductor with Spark. The examples in this procedure use `rg_shared` to refer to the resource group, but you can specify any name.

2. Create a Spark instance group.
   a. Make sure that only Spark executors and the shuffle service use the shared resource group `rg_shared`.
      For more information about creating Spark instance groups, see [http://www-01.ibm.com/support/knowledgecenter/SSVH2B_1.1.0/developing Instances/developing Instances.dita](http://www-01.ibm.com/support/knowledgecenter/SSVH2B_1.1.0/developing Instances/developing Instances.dita)
   b. Start the instance group and wait for its deployment to finish.

3. Change shuffle service to a different resource group.
   After the Spark instance group starts, both the Spark executors and the shuffle service run on the shared resource group `rg_shared`. To prevent potential conflicts with EGO, you must configure the shuffle service to run on another resource group.
a. Create a resource group for the shuffle service with the same configuration and host set as the first shared resource group. For example, create a new resource group `rg_service` with the same configuration and host set as `rg_shared`.

b. Find all consumers that have access to `rg_shared` and grant access permission to `rg_service` as well.

c. Switch the shuffle service to run on the new resource group `rg_service`.
   1) From the management console, select **Spark Instance Groups**.
   2) Stop the running instance group that you created.
   3) Select the stopped Spark instance group and click the **Services** tab.
   4) Click a service with the description "Spark Shuffle Service" and click **"Modify Service Profile"**.
   5) Click **Save** and start the instance group.
   6) Change the value of the **ego:ResourceGroupName** property to `rg_service`. Make sure that any changes you make to `rg_shared` are repeated with `rg_service`.

4. Create consumers to be used by LSF to borrow hosts.

   To create new consumers for LSF at the right level of the existing consumer tree, see [http://www.ibm.com/support/knowledgecenter/SSVH2B_1.1.0/shared_files/asc_consumers.dita](http://www.ibm.com/support/knowledgecenter/SSVH2B_1.1.0/shared_files/asc_consumers.dita)

   **Note:** When you specify the LSF consumer properties, make sure to do the following steps.
   - Specify all shared resource groups (for example, `rg_shared` and `rg_service`) that the LSF consumer needs to have access to.
   - Specify a reclaim grace period to impose a delay before LSF daemons are shut down after the `egosh` `ego restart all` command reclaims the host.

   For example, if you create two new consumers (`TmplA_LSFConsumer` and `TmplB_LSFConsumer`), which are siblings to existing IBM Spectrum Conductor with Spark consumers (`Conductor_ConsumerA` and `Conductor_ConsumerB`), the following consumer tree is used:

   ```
   Tmpl1A_Consumer
   \-Conductor_ConsumerA
   \-Tmpl1A_LSFConsumer
   Tmpl1B_Consumer
   \-Conductor_ConsumerB
   \-TmplB_LSFConsumer
   ```

5. Enable borrow-only consumers in the IBM Spectrum Conductor with Spark configuration.

   a. Log in to the IBM Spectrum Conductor with Spark master host as root.
   b. Set the following parameter in the `$EGO_CONFDIR/ego.conf` file:

   ```
   EGO_ENABLE_BORROW_ONLY_CONSUMER=Y
   ```

6. Restart IBM Spectrum Conductor with Spark for the change to take effect.

   ```
   egosh ego restart all
   ```

7. Modify the resource plan.

   After the new LSF consumers are created, modify resource plans for shared resource groups (for example, `rg_shared` and `rg_service`) to allocate resources among IBM Spectrum Conductor with Spark consumers and LSF consumers.
For more information about modifying resource plans, see [http://www.ibm.com/support/knowledgecenter/SSVH2B_1.1.0/shared_files/resource_plan_creating.dita](http://www.ibm.com/support/knowledgecenter/SSVH2B_1.1.0/shared_files/resource_plan_creating.dita).

Set the priority of LSF consumers to be lower than IBM Spectrum Conductor with Spark consumers to force giving up all its borrowed resources when a sibling IBM Spectrum Conductor with Spark consumer has demand. In the resource plan, specify the appropriate ratio and priority for LSF consumers.

a. Specify slot allocation policy to **Stacked**.

b. For share ratio, specify 0 for TmplA_LSFConsumer and 1 for Conductor_ConsumerA.

c. For consumer rank, specify 100 for TmplA_LSFConsumer and 0 for Conductor_ConsumerA.

---

**Installing LSF on IBM Spectrum Conductor with Spark compute hosts**

Follow these steps to install LSF on IBM Spectrum Conductor with Spark compute hosts.

The examples in this procedure use `conductorhost` to refer to this resource, but you can specify any name. For more information about adding resources, see "Adding new resources to your cluster" in *Administering IBM Spectrum LSF*.

1. Identify shared hosts.

   Find the shared resource groups to be used by the resource connector (for example, `rg_shared` and `rg_service`), and identify all shared hosts in resource group.

   For more information about viewing hosts in a resource group in IBM Spectrum Conductor with Spark, see [Viewing hosts in a resource group](#).

   For shared hosts with LSF, you can install LSF can be installed locally or on a shared directory. However, the shared directory must not be the same as the LSF cluster's shared directory. Shared hosts can be configured statically in the LSF cluster file, or configured to join the LSF cluster dynamically when they first start. This procedure gives the steps for dynamic hosts.

   On each shared host, follow the steps in *Installing IBM Spectrum LSF on UNIX and Linux* to install LSF.

2. Log in as root.

3. Configure DNS.

   a. Configure the DNS settings to ensure that each shared host is able to look up host name and address of LSF master and all LSF compute hosts.

   b. Update environment settings for the IP address conversion. Update environment settings for the IP address conversion.

   The DNS configuration mechanism might be the `/etc/hosts` file, the DNS service, or another setting, depending on your LSF environment.

4. Edit the `slave.config` file to set the installation options.

   For example, specify the following parameters in the `slave.config` file:

   ```
   LSF_TOP="/usr/share/lsf"
   LSF_ADMIN="lsfadmin root"
   LSF_TARDIR="/opt/lsfinstaller/"
   LSF_CLUSTER_NAME="cluster1"
   LSF_MASTER_LIST="hostm hostd"
   LSF_ENTITLEMENT_FILE="/opt/lsfinstaller/lsf_std_entitlement.dat"
   LSF_SERVER_HOSTS="hosta" # LSF master host
   LSF_LIM_PORT="17869" # master LIM port
   LSF_LOCAL_RESOURCES="[resource conductorhost]"
   ```

Chapter 3. Configuring IBM Spectrum Conductor with Spark for LSF resource connector 17
5. Use the `lsfinstall` script to install LSF on the compute node.

```
lsfinstall -s -f slave.config
```

For more information about installing LSF, see "Installing a new cluster" in *Installing IBM Spectrum LSF on UNIX and Linux*.

6. Create LSF administrative scripts.

Create two scripts that the resource connector calls to start and stop LSF daemons on shared hosts as they are allocated and reclaimed. The scripts must have read and execute permissions for the root user.

The following example is a script for starting LSF daemons. Change `LSF_TOP` to your actual installation directory.

```
cat /usr/share/lsf/lsf.start
#!/bin/sh
LSF_TOP=/usr/share/lsf
source $LSF_TOP/conf/profile.lsf
lsadmin limstartup
lsadmin resstartup
badmin hstartup
```

The following example is a script for stopping LSF daemons. Change `LSF_TOP` to your actual installation directory.

```
cat /usr/share/lsf/lsf.stop
#!/bin/sh
LSF_TOP=/usr/share/lsf
source $LSF_TOP/conf/profile.lsf
badmin hshutdown
lsadmin resshutdown
lsadmin limshutdown `hostname`
```

---

**Configuring LSF resource connector for IBM Spectrum Conductor with Spark**

Specify LSF resource connector configuration to enable IBM Spectrum Conductor with Spark as a resource provider.

1. Update the IBM Spectrum Conductor with Spark connection configuration.

Change the following parameters in the `egoprov_ego.conf` file to enable the resource connector to connect to the IBM Spectrum Conductor with Spark cluster:

- `EGO_MASTER_LIST`  
- `EGO_LIM_PORT`  
- `EGO_KD_PORT`  
- `EGO_PEM_PORT`  
- `RC_EGO_VERSION`

2. Update resource connector administration configuration. Change the following parameters in the `egoprov_config.json` file to enable the resource connector to manage the IBM Spectrum Conductor with Spark cluster:

- `Username`  
- `Password`  
- `StartLSFCmd`  
- `StopLSFCmd`  
- `GracePeriod`  
- `EGOSharedHostsResourceGroup`

3. Create templates. Create at least one template in the `egoprov_templates.json` file.
Make sure that your template accurately defines at least the following attributes:

- ncpus
- conductorhost

The EGO_select string must match shared hosts in the rg_shared resource group. The following template is a minimal example for hosts with 4 CPUs:

```json
{
    "Templates": [
        {
            "Name": "TemplateA",
            "Attributes": {
                "ncpus": ["Numeric", "4"],
                "conductorhost": ["Boolean", "1"]
            },
            "EGO_select": "select(ncpus==4)"
        }
    ]
}
```
Chapter 4. Configuring OpenStack for LSF resource connector

Follow these steps to configure OpenStack to create instances for LSF resource connector to make allocation requests on behalf of LSF.

- You must have root access to the LSF master host.
- The LSF cluster must have a DNS server.
- You must have correct permissions to update the DNS server.
- You must be able to restart the LSF cluster.
- **OpenStack Liberty is already installed.** For installation steps, see [docs.openstack.org/liberty/](docs.openstack.org/liberty/)
- You must be familiar with and have the ability to perform OpenStack administrative operations.
- The virtual network to be used by OpenStack virtual instances must be configured so that they can communicate with LSF hosts.

**Important:** The following OpenStack services are required for LSF resource connector with OpenStack as the resource provider:
- Identity service (Keystone)
- Image service (Glance)
- Compute service (Nova)
- Networking service (Neutron)

LSF resource connector has been tested on the following systems:
- Linux2.6-glibc2.3-x86_64
- LSF 9.1.3 and 10.1 Standard Edition
- OpenStack Liberty, tested on CentOS 7

In the following steps, you must perform all operations as the OpenStack administrator unless otherwise stated.

1. Create projects, users, and roles for LSF. This step does not require OpenStack administrator privileges. Create a non-administrator project, user, and role for LSF. The LSF user must have access to resources such as images and networks created by the administrator.

2. Modify security group for LSF. Add rules to open all LSF listening ports to the security groups that are used to launch instances. The ports must match those from the existing LSF cluster. The following are the default port number values:
   - **LSF_LIM_PORT=7869** (TCP and UDP)
   - **LSF_RES_PORT=6878** (TCP)
   - **LSB_SBD_PORT=6882** (TCP)

3. Build an LSF cloud image. The LSF cloud image is a single file that contains a virtual disk image with the following components installed:
   - Bootable Linux2.6-glibc2.3-x86_64
   - LSF 9.1.3 or 10.1 Standard Edition installed
   - **cloud-init** installed
     a. Create the virtual machine image. For more details on downloading an official image, refer to the following information (for example,
b. Launch an instance with the image. LSF installation requires root access in the OpenStack instances. However, the official images do not have root enabled. Therefore, you must enable root when launching the OpenStack instances. The following is an example of a userdata script for the CentOS image. Launch an instance with this script to enable root.

```
#!/bin/bash

if ![ ! -f /etc/sudoers.d/90-cloud-init-users ]; then
    ( cat <<EOF
    # User rules for centos
centos ALL=(ALL) NOPASSWD:ALL
    EOF
    ) > /etc/sudoers.d/90-cloud-init-users
    chmod 440 /etc/sudoers.d/90-cloud-init-users
else
    echo "/etc/sudoers.d/90-cloud-init-users exists" > /root/cloud-init-user.log
    cat /etc/sudoers.d/90-cloud-init-users >> /root/cloud-init-user.log
fi
```

c. Configure the instance.

1) Log in to the instance.
   Log in to the instance as the default cloud user and switch to root with the `sudo su -` command.

2) Set up firewall communications.
   Modify the instance's firewall to open all LSF listening ports. The ports must match those from the existing LSF cluster. The following are the default port number values:
   - `LSF_LIM_PORT=7869` (TCP and UDP)
   - `LSF_RES_PORT=6878` (TCP)
   - `LSB_SBD_PORT=6882` (TCP)

d. Install LSF on the instance.

1) Log in to the instance.
   Log in to the instance as the default cloud user and switch to root with the `sudo su -` command.

2) Install LSF with parameters specified in the `slave.config` file.
   For example, specify the following parameters in the `slave.config` file:
   ```
   LSF_TOP="/usr/share/lsf"
   LSF_ADMINS="lsfadmin root"
   LSF_CLUSTER_NAME="cluster1"
   LSF_MASTER_LIST="hostm hostd"
   LSF_TARDIR="/opt/lsfinstaller/"
   LSF_ENTITLEMENT_FILE="/opt/lsfinstaller/lsf_std_entitlement.dat"
   LSF_SERVER_HOSTS="hostm" # LSF master host
   LSF_LIM_PORT="17869" # master LIM port
   LSF LOCAL_RESOURCES="[resource openstackhost]"
   ```
   Use the `lsfinstall -s -f slave.config` command to install the LSF on the instance.

   For full details on installing LSF, see *Installing IBM Spectrum LSF on UNIX and Linux*.

3) Prepare users for LSF.
   To get the job submitted by a user to run on the instance, the instance must have this user prepared or LSF user mapping configured. For more
information about user groups and user account mapping, see
"Managing Users and User Groups" and "Between-Host User Account
Mapping" in Administering IBM Spectrum LSF.

4) Log out from the instance.

  e. Snapshot the instance.
Create snapshot for the instance and ensure that this newly-created image
can be accessed by LSF projects.

---

Configuring the DNS server for OpenStack

Set up an external DNS server for OpenStack instances.

1. Specify a Linux host as the OpenStack DNS server. This host can be a controller
   node or another Linux host.
2. Log on to the OpenStack DNS server as root.
3. Copy the install_dns.sh script file to the OpenStack DNS server. The script is
   located in $LSF_TOP/<LSF_VERSION>/resource_connector/openstack/scripts/
   install_dns.sh.
   a. Change the following parameters
      - VAR_OPENSTACK_DOMAIN=<instance_domain>
        The instance domain must be the same as the dns_domain parameter in
        the OpenStack configuration file (/etc/neutron/neutron.conf).
      - VAR_OPENSTACK_INSTANCE_PREFIX=<instance_prefix>
        The instance prefix must be the same as the "InstancePrefix" parameter in
        the osprov_config.json file.
      - VAR_OPENSTACK_SUBNET_CIDR=<subnet_cidr>
        The public subnet for virtual instances.
      - VAR_OPENSTACK_DNS_ADDRESS=<openstack_dns_address>
      - VAR_LSFS_DOMAIN=<lsf_domain>
      - VAR_LSFS_REVERSE=<lsf_reverse_query>
      - VAR_LSFS_DNS_ADDRESS=<lsf_dns_address>
      For example,
      VAR_OPENSTACK_DOMAIN=openstack.vm
      VAR_OPENSTACK_INSTANCE_PREFIX=host
      VAR_OPENSTACK_SUBNET_CIDR=10.110.135.0/26
      VAR_OPENSTACK_DNS_ADDRESS=10.110.135.210
      VAR_LSFS_DOMAIN=lsf.domain
      VAR_LSFS_REVERSE=54.42.10
      VAR_LSFS_DNS_ADDRESS=10.42.54.77
      b. Run the script to set up a DNS server for OpenStack instances.
4. Configure the LSF DNS server. To enable LSF hosts to resolve instance domain
   names, add the DNS forward zone for the instance domain to the DNS server.
The following steps assume that you are using Linux bind as the DNS server:
  a. Log on to the LSF DNS server as root.
  b. Disable the empty-zone feature. Edit the configuration file /etc/named.conf
to disable empty-zone:

      options {
          ... 
          empty-zones-enable no;
          ... 
      };
c. Add a forward zone for the virtual instance domain. Edit the configuration file /etc/named.conf to add forward zones in the following format:

```plaintext
zone "<instance_domain>" IN {
    type forward;
    forwarders { <openstack_dns_server> ;
    }
};
zone "<instance_reverse_zone>" IN {
    type forward;
    forwarders { <openstack_dns_server> ;
    }
};
```

A typical configuration is shown in the following example:

```plaintext
zone "openstack.vm" IN {
    type forward;
    forwarders { 10.110.135.210; }
};
zone "135.110.10.in-addr.arpa" IN {
    type forward;
    forwarders { 10.110.135.210; }
};
```

d. Restart the DNS server.

```
  service named restart
```

---

**Configuring LSF resource connector for OpenStack**

Specify LSF resource connector configuration to enable OpenStack as a resource provider.

1. Update the OpenStack administrative configuration. Change the parameters in the osprov_config.json file to enable the resource connector to connect to OpenStack.

2. Create templates. Create at least one template in the osprov_templates.json file.

   Make sure that your template accurately defines at least the following attributes:
   
   - ncpus
   - openstackhost

   The following template is a minimal example for hosts with 4 CPUs:

   ```json
   {
     "Templates": [
       {
         "Name": "TemplateA",
         "Attributes": {
           "ncpus": ["Numeric", "4"],
           "openstackhost": ["Boolean", "1"]
         },
         "Image": "CentOS-6.x-LSF-x86_64",
         "Flavor": "m1.large",
         "MaxNumber": "10",
       }
     ]
   }
   ```

3. Optional. Add additional parameters to the OpenStack create server requests. Resource connectors provide an extension point to customize parameters for create server request. Edit the create_server.json file in the
Enter parameters in JSON format as described in the OpenStack compute API (HTTP POST /v2.1/[tenant_id]/servers): [developer.openstack.org/api-ref-compute-v2.1.html](developer.openstack.org/api-ref-compute-v2.1.html)

For example, specify the availability zone to "nova" for all instances:

```json
{
    "server": {
        "availability_zone": "nova"
    }
}
```
Chapter 5. Configuring Amazon Web Services for LSF resource connector

Follow these steps to configure Amazon Web Services (AWS) for LSF resource connector to make allocation requests from AWS on behalf of LSF. LSF clusters can launch instances from AWS to satisfy pending workload. The instances join the LSF cluster. If instances become idle, LSF resource connector terminates them.

- You must have root access to the LSF master host.
- Both the LSF master host and the compute nodes (AWS instances) must be reachable from each other.
- You must be able to restart the LSF cluster.
- You must be familiar with and be able to administer AWS.
- The virtual network to be used by AWS virtual instances must be configured so that they can communicate with LSF hosts.
- You must configure the AWS instances to map users to the LSF cluster submission users. For example, add the submission users to the AWS instance or synchronize the users on the launched AWS instances.
- Decide how you want to authenticate AWS users: you can use either an IAM user or a federated account to access AWS.

**Note:** Both IAM user credentials and federated accounts support all AWS RC features. The user for either option must be assigned the correct permissions.

LSF resource connector was tested on the following systems:
- LSF 10.1 master host on Linux x86 Kernel 3.10, glibc 2.17 RHEL 7.x
- Instances on Linux x86 Kernel 3.10, glibc 2.17 CentOS 7.x

LSF resource connector is assumed to work on the following systems:
- IBM Spectrum LSF 10.1
- Linux x86 Kernel 2.6, glibc 2.5 RHEL 5.x
- Linux x86 Kernel 2.6, glibc 2.11 RHEL 6.x
- Linux x86 Kernel 3.0, glibc 2.11 SLES 11.x
- Linux x86 Kernel 3.11, glibc 2.18 SLES 12.x
- Linux x86 Kernel 4.4, glibc 2.23 Ubuntu 16.04 LTS

**Important:** In the following steps, you must perform all operations as the AWS administrator unless otherwise stated.

1. Choose whether to use an IAM user or federated account to access AWS or a federated account script.
   - Create AWS Identity and Access Management (IAM) users and roles for LSF. IAM allows secure access to AWS resources for users and also allows shared access to an AWS account.
   - This step does not require AWS administrator privileges.
   - Create a non-administrator user and role for LSF. The LSF user must have access to resources such as images and networks that are created by the administrator.
   - OR
• Create a federated account script.

Resource connector also supports federated accounts for LSF resource connector as an option instead of requiring permanent AWS IAM account credentials. Federated users are external identities that are granted temporary credentials with secure access to resources in AWS without requiring creation of IAM users. Users are authenticated outside of AWS (for example, through Windows Active Directory).

For more information about configuring LSF resource connector to use federated accounts, see “Configuring AWS access with federated accounts” on page 32.

Important: To use existing enterprise users stored in Active Directory (federated accounts) you must use temporary credentials.

To access AWS, you must obtain either a permanent AWS IAM credential or a set of temporary credentials.

Resource connector gets user role information for AWS instances two ways:
• Create an IAM access key and credential files.
• Create roles and an instance profile to get temporary credentials for use in an AWS template.

Complete the following steps to create an access key and credential files:

a. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/

IAM allows secure access to AWS resources for users and also allows shared access to an AWS account.

If you create an access key for each user using the web GUI, you must download the credentials. A credentials.csv file is generated. Provide the access key ID and secret access key in the file to LSF for authentication.

b. Copy the access key ID and the secret access key to $LSF_ENVDIR/resource_connector/aws/conf/credentials or any other location that is accessible by LSF and specify the path for it in the awsprov_config.json file.

Fast path: The aws_enable.sh sets aws_access_key_id and aws_secret_access_key in the credentials file with the value set for the AWS_IAM_CREDENTIAL_ID and AWS_IAM_CREDENTIAL_KEY parameters in the aws_enable.config file.

The credentials file must be in the following format:

```
cat credentials
[default]
aws_access_key_id=AKIAIDZWNW2Q
aws_secret_access_key=/pYgqypCzJedT92dENPq74Bab
```

Credentials that you generate by using the AWS command-line interface must also be in this format.

c. Provide the path of the key file of the created AWS key pair and the name of the key file.

Fast path: The aws_enable.sh file uses the values from the for the AWS_REGION, AWS_SSH_KEY_PATH, and AWS_FED_CREDENTIAL_SCRIPT parameters in the aws_enable.config file to update the awsprov_config.json file.

The path of the key file is in awsprov_config.json, and the name of the key file is in awsprov_templates.json to allow different templates to use different keys.
AWS uses public-key cryptography to secure the login information for an instance. A Linux instance has no password; you use a key pair to log in to your instance securely. You specify the name of the key pair when you launch your instance, then provide the private key when you log in using SSH.


Provide the key file and key name to LSF to launch Amazon Elastic Compute Cloud (EC2) instances. The private key must be provided to log in by using SSH.

Provide the path of the key file and the name of the key file in the `awsprov_templates.json` file.

Complete the following steps to create an instance profile for your AWS instances. The instance profile uses the permissions provided by a specific user role without needing to include keys in an on-disk credentials file:

a. Sign in to the AWS Management Console and open the IAM console at [https://console.aws.amazon.com/iam/](https://console.aws.amazon.com/iam/)

b. Open the IAM section in the AWS console and choose **Roles** in the navigation bar.

c. Click **Create New Role** and follow the prompts.

IAM roles group AWS access control privileges together. A role can be assigned to an IAM user or an IAM instance profile. IAM Instance Profiles are containers for IAM roles that allow you to associate an EC2 instance with a role through the profile. The EC2 runtime environment contains temporary credentials that have the access control permissions of the profile role.

d. Use the IAM page in the AWS console to create an instance profile that contains these permissions and assign it to the AWS user.

The AWS user ID that that creates instances with assigned IAM roles must have at least the following permissions granted:

- `ec2:RunInstances`
- `ec2:AssociateIamInstanceProfile`
- `ec2:ReplaceIamInstanceProfileAssociation`
- `iam:PassRole`

**Note:** You might need to assign the following policies to the role:

- `AmazonEC2FullAccess`
- `AmazonVPCFullAccess`

e. Create an instance profile and assign a role to it.

An instance profile maps an AWS EC2 compute instance to an IAM role. The instance profile short name is equivalent to the role name, and the instance profile ARN appears in the corresponding field on the role property page.

When you create a role through the AWS console, creation of a role automatically creates an instance profile associated with the role.

f. Request an EC2 instance and specify the instance profile to verify the role.

Applications running in the instance can obtain temporary credentials associated with the profile role.

To make the roles available for resource connector to create instances, specify the IAM roles that you created in an AWS resource template in the
awsprov_templates.json file. Resource connector uses that information to request EC2 compute instances with particular instance profiles. Jobs that run on those hosts use temporary credentials provided by AWS to access the AWS resources that the specified role has privileges for.

2. Optional: Create a security group for LSF.
You can use the default security group that is provided by AWS or create your own with customized rules to open all LSF listening ports to the security groups that are used to launch instances. The ports must match the ports in the existing LSF cluster.
LSF has the following default port number values:
  • LSF_LIM_PORT=7869 (TCP and UDP)
  • LSF_RES_PORT=6878 (TCP)
  • LSB_SBD_PORT=6882 (TCP)
You can also add master host IP address to accept all traffic from the master host.

Note: If you allow only the traffic from the default ports, some nios commands might not work, since the ports are configured for them dynamically and are different from the default ports.

3. Create a Virtual Private Cloud (VPC) and subnets.
A Virtual Private Cloud (VPC) is a virtual network that is dedicated to an AWS account.
A subnet is a range of IP addresses in your VPC. You can launch AWS resources into a subnet that you select. Use a public subnet for resources that must be connected to the internet, and a private subnet for resources that aren't connected to the internet.

If the Auto-Assign Public IP option is checked, public IP or public DNS is available for created instances in this subnet. Auto-Assign Public IP is required if you want to SSH to the instance created.

By default, Amazon assigns a private DNS to the instances that are created. If your master host is also running on AWS, then you don't need to enable the Auto-Assign Public IP because all LSF hosts are on the same network, and the master host and slave hosts are reachable from each other.

In most of the cases, the master host does not run on AWS. Slave hosts must have a public IP address for the master LIM to connect to the slave LIMs.

You must specify the name of the subnet (subnetId) in the awsprov_templates.json file that is used to launch the instance.

4. Build the LSF cloud image.
To create an Amazon Machine Image (AMI) for an LSF cloud compute host, the cluster administrator must first manually launch an instance and install LSF on an EC2 instance. An image is then created that uses this instance. Subsequent cloud instances can be dynamically launched by using the AMI.

After the instance is created, use ssh to log in to the instance with the key file you generated.

a. Copy the LSF package to EC2 host.
  • lsf10.1_linux2.6-glibc2.3-x86_64.tar.Z
b. Set up firewall communications on the instance.
Modify the instance’s firewall to open all LSF listening ports. The ports must match those from the existing LSF cluster. The following are the default port number values:

- **LSF_LIM_PORT=7869** (TCP and UDP)
- **LSF_RES_PORT=6878** (TCP)
- **LSB_SBD_PORT=6882** (TCP)

a. Prepare users for LSF.
To get the job submitted by a user to run on the instance, the instance must have this user prepared or LSF user mapping configured. For more information about user groups and user account mapping, see “Managing Users and User Groups” and “Between-Host User Account Mapping” in *Administering IBM Spectrum LSF*.

a. Install the software.
ed.x86_64
If no software is installed, you can use command `yum install` to install it.
yum install ed

b. Install LSF as a slave host on AWS EC2.
`./lsfinstall -s -f slave.config`
The following example shows typical installation parameters to set in the `slave.config` file.

```
cat /home/ec2-user/lsf/lsf10.1_lsfinstall/slave.config
LSF_TOP="/home/ec2-user/lsf"
LSF_ADMINS="ec2-user"
LSF_TARDIR="/home/ec2-user/lsf/"
LSF_LICENSE="/home/ec2-user/lsf/lsf_std_entitlement.dat"
LSF_SERVER_HOSTS="master.myserver.com"
LSF_LOCAL_RESOURCES="[resource awshost] [resource define_ncpus_threads]"
LSF_LIM_PORT="7869"
```

**LSF_LIM_PORT**
The port number must be the same as the one defined on the LSF master host of your cluster.

**LSF_GET_CONF**
Updates the LSF configuration to synchronize the cluster configuration with the master host.

**LSF_LOCAL_RESOURCES**
LSF uses the Boolean resource name `awshost` to identify AWS instances. Instances that are used by LSF are omitted from `bhosts` output unless you specify the `-a` option.

**Note:** By default, LSF maps `ncpus` to cores (`number_processors x number_cores`), but AWS treats each virtual CPU as a hyperthreaded core. To map the exact number of AWS instance virtual CPUs to LSF `ncpus`, add the resource name `define_ncpus_threads` to the list of local resources. The `define_ncpus_threads` resource maps the number of LSF `ncpus` to threads instead of cores for AWS slave hosts.

c. If required, update the `/etc/hosts` file to add the master host name to the `/etc/hosts` file on the EC2 host.

d. Start the LSF daemons on the instance manually and make sure that the instance can join the master host cluster as a dynamic host.
If the instance cannot join the cluster, check the VPN, firewall, or security group settings. Check whether the master host can ping the instance by using its public IP address and vice versa. If the master host can ping the instance by using its IP address but not by using the host name, DNS settings need to be configured.

e. Shut down the daemons and log out of the instance.

f. Snapshot the instance.

Create an Amazon machine image (AMI) for the instance.

The name of the image (imageId) is required in the awsprov_templates.json file for LSF to decide when borrowing happens, and which instances are launched from which image.

---

**Configuring AWS access with federated accounts**

Resource connector supports federated accounts for LSF resource connector as an alternative to requiring permanent AWS IAM account credentials. Federated users are external identities that are granted temporary credentials with secure access to resources in AWS without requiring creation of IAM users. Users are authenticated outside of AWS (for example, through Windows Active Directory). All AWS resource connector features are supported when you use federated accounts instead of IAM credentials.

The LSF administrator must create a script or executable that can be executed by the primary LSF administrator to generate temporary credentials to be used with AWS. The temporary credentials also must have the assigned role that has the policies associated for AWS permissions required by LSF.

The script must be accessible from LSF master candidate hosts. The script must also be set for the default profile. The script must output to stdout in the following format.

```
[default]
aws_access_key=<value>
aws_secret_access_key=<value>
aws_session_token=<value>
```

For example:

```
[default]
aws_access_key=aGJ3P1gFRQCEsPpTSeq=fOTLqSisLcH1dPqm
aws_secret_access_key=ASIAJ6UWHCMWUWRECOKIQ
aws_session_token=FQoDYXdZEnr/人身/wdJbI1TEA7sIj9lSUwN6z2DyqjHhSNIHJUCOsW+pZ8Ww+CBcNNTN3InLMUBPn8zj
ItmKnniPMFbwZn=qfQCCl/0jaNgKXzUdpfVhe02G6Gr8bZno4Zgy7OmiTTugiuUTBh9YKK270BZHi86jZvAV0v2mbFQaznWVhKq2slMSy7JC4bmAFPNCN8lWfkY7AVbYtwTnxFp6peVS2dergSH1ef9nU+V9WW7n0yZlyOo+iXgU=
```

The script is executed automatically by LSF when a credential is required to access AWS services. The script cannot be interactive since it will be run automatically by LSF.

**Note:** If an Active Directory user name and password is required for the script, it must be done automatically by storing it in environment variables or a secure file. The environment variable or file must be readable by the LSF primary administrator because that user executes the script.

Identity federation in AWS allows external identities (federated accounts) to access AWS services and resources while being authenticated and authorized outside of AWS.
AWS IAM. This allows integration of AWS with existing authentication services (for example, Active Directory) in enterprise environments instead of requiring administrators to create IAM user credentials for each existing user.

**Important:** To use existing enterprise users stored in Active Directory (federated accounts) you must use temporary credentials.

Federated accounts in AWS require temporary credentials to be generated to allow connection to AWS. These credentials are temporary and expire after a specific duration. LSF resource connector generates the temporary credentials through execution of an administrator-defined script or executable that is able to generate the temporary credentials for LSF to use. LSF uses the temporary credentials to establish a connection to AWS to launch, monitor and terminate EC2 instances.

Use the `AWS_CREDENTIAL_SCRIPT` parameter in the `awsprov_config.json` file to specify a path to the script that generates temporary credentials for federated accounts. For example,

`AWS_CREDENTIAL_SCRIPT=/shared/dir/generateCredentials.py`

LSF executes the script as the primary LSF administrator to generate a temporary credentials before it creates the EC2 instance.

**Configuring LSF resource connector for Amazon Web Services**

Specify LSF resource connector configuration to enable Amazon Web Services (AWS) as a resource provider.

1. Update the AWS administrative configuration. Change the following parameters in the `awsprov_config.json` file to enable the resource connector to connect to AWS:
   - `AWS_CREDENTIALS_FILE` or `AWS_CREDENTIALS_SCRIPT`
   - `AWS_REGION`
   - `AWS_KEY_FILE`

   The `AWS_CREDENTIALS_FILE`, `AWS_CREDENTIALS_FILE`, `AWS_REGION`, and `AWS_KEY_FILE` are AWS user credentials that are created in Chapter 5, “Configuring Amazon Web Services for LSF resource connector,” on page 27.

2. Create AWS instance templates. Create at least one template in the `awsprov_templates.json` file.

   Make sure that your template accurately defines at least the following attributes:
   - `ncpus`
   - `awshost`

   The following template is a minimal example for hosts with 4 CPUs:

   ```json
   {
   "Templates": [
   {
   "templateId": "TemplateA",
   "attributes": {
   "ncpus": ["Numeric", "4"],
   "awshost": ["Boolean", "1"]
   },
   "imageId": "ami-27a1",
   "vmType": "t2.micro",
   "subnetId": "subnet-b573",
   ```
3. Optional: Update the location of the LSF_TOP directory in the user_data.sh script to start LSF daemons on the instance. This script runs during the AWS instance startup.

The user_data.sh script is located in the `<LSF_TOP>/<LSF_VERSION>/resource_connector/aws/scripts` directory.

**Note:** You can also add any extra code that needs to be run at the time of the AWS instance launch.

For example, specify the directory where LSF is installed on the Amazon Machine Image:

```bash
#!/bin/bash

LSF_TOP=/usr/share/lsf
source $LSF_TOP/conf/profile.lsf
lsadmin limstartup
lsadmin resstartup
badmin hstartup
```

**Configuring user scripts to register Amazon Web Services hosts with the LSF master host**

If a DNS server is not set up to resolve the LSF master host and AWS instances, use the user_data.sh script under the `<LSF_TOP>/<LSF_VERSION>/resource_connector/aws/scripts` directory to register the dynamic hosts and resolve the DNS entries on both sides.

Set up the user_data.sh script to use a public DNS to resolve an EC2 public host name and IP address to communicate with the LSF master host directly.

LSF looks up host names and addresses for all communication between AWS instances and master host candidates. Make sure that your environment settings for IP address resolution work between the LSF master host and instances that join the cluster.

Check whether the master host candidates can ping the instance by using both its public IP address and and the host name reported by the `hostname` command and vice versa. Make sure that the instances can ping each other with both public IP address and reported host name. If the instances can ping each other only by using IP address but not by using the host name, DNS settings need to be configured.

If the instance cannot join the cluster, check the VPN, firewall, or security group settings. You must open firewall rules for all LSF ports between the LSF master and borrowed slave host IP ranges.

**Fast path:** If you use the aws_enable.sh script and the aws_enable.config file to automate AWS setup, the user_data.sh file under `<LSF_TOP>/<LSF_VERSION>/resource_connector/aws/scripts` is automatically updated:

- The path to LSF_TOP is set in the script so that the script can be executed normally during AWS instance startup. The LSF_TOP set to the value of the AWS_LSF_TOP parameter in the aws_enable.config file.
The script runs the **lsreghost** command in the `user_data.sh` script to resolve the master host in the AWS instance. If the DNS server is not set up to resolve the master host and AWS instances, the DNS entry is resolved by running the **lsreghost**.

1. Set the correct path for `LSF_TOP`, where LSF is installed on the slave host.
   ```
   LSF_TOP=/home/ec2_user/lsf
   ```
2. Source LSF environment.
   ```
   source $LSF_TOP/conf/profile.lsf
   ```
3. Set the master host name in the `hostregsetup` script.
   ```
   echo "master.myserver.com" > $LSF_ENVDIR/hostregsetup
   ```
4. Run the **lsreghost** script to resolve the DNS entry.
   ```
   lsreghost -s $LSF_ENVDIR/hostregsetup
   ```

When AWS creates an instance the EC2 host has two IP addresses and two host names (internal and public). By default, AWS uses the internal IP address and host name that is reported by the `hostname` command to communicate, but an external (public) LSF master host cannot recognize the private host name and IP address. For the master host to recognize the instance and use public DNS to resolve the EC2 public host name and public IP address of the instance, you must update the internal host name of the instance to its public host name before it joins the LSF cluster.

To use a public DNS to resolve an EC2 public host name and IP address to communicate with the LSF master host directly, add the following code to the `user_data.sh` script give a machine host name that the master host can recognize:

```
# Get my public IP
publicIP=$(curl whatismyip.akamai.com)
# Make up AWS EC2 host name with public DNS
publicName=ec2-$(publicIP//.)/us-west-2.compute.amazonaws.com
# Update host name
echo ${publicName} > /etc/hostname
hostname ${publicName}
```

### Use AWS Spot instances

**Use Spot instances** to bid on spare Amazon EC2 computing capacity. Since Spot instances are often available at a discount compared to the pricing of On-Demand instances, you can significantly reduce the cost of running your applications, grow your application's compute capacity and throughput for the same budget, and enable new types of cloud computing applications.

With Spot instances you can reduce your operating costs by up to 50-90%, compared to on-demand instances. Since Spot instances typically cost 50-90% less, you can increase your compute capacity by 2-10 times within the same budget.

Spot instances are supported on any Linux x86 system that is supported by LSF.


### Requesting Spot instances

Submit the job that requires Spot instance pricing with the `pricing==spot` resource requirement in the **bsub** command:
bsub -R "awshost && pricing=spot" mycovfefe

The pricing resource must be configured in the lsf.shared file:

```
Begin Resource
RESOURCENAME      TYPE     INTERVAL      INCREASING   DESCRIPTION
...                pricing   String       ()          ()       (Pricing option: spot/ondemand)
...                End Resource
```

Spot instances are reclaimed when the Spot price goes higher than the current bid price.

You can also configure an AWS template to use Spot instances.

```
awsprov_templates.json:
{
    "templateId": "aws-spotvm-demo",
    "maxNumber": 2,
    "attributes": {
        ...,
        "awshost": ["Boolean", "1"],
        "pricing": ["String", "spot"],
        },
        ...
        ...
        ...
    "userData": "pricing=spot"
}
```

Edit the user_data.sh script to use the Spot instance pricing resource:

```
#!/bin/bash

echo START >> /var/log/user-data.log 2>&1
# run hostsetup
...
if [ -n "$pricing" ]; then
    sed -i "s/\(LSF_LOCAL_RESOURCES=.*\)"/
    [resourcemap $pricing]*pricing]/" $LSF_CONF_FILE
    echo "update LSF_LOCAL_RESOURCES lsf.conf successfully, add [resourcemap $pricing]*pricing]" >> $logfile
fi...
```

The user_data.sh script is located in the `<LSF_TOP>/LSF_VERSION>/resource_connector/aws/scripts` directory.

**Security requirements for Spot instances**

- You must create a Spot fleet role and add the corresponding Amazon Resource Name (ARN) to the awsprov_templates.json template configuration file. For steps to create a Spot fleet role, see [http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/spot-fleet-requests.html#spot-fleet-prerequisites](http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/spot-fleet-requests.html#spot-fleet-prerequisites)

- The AWS user linked to the access key that is stored in the credentials file must have the Spot fleet permissions to bid on, launch, and terminate the configured Spot fleets. For steps to add permissions to a user, see [http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/spot-fleet-requests.html#spot-fleet-prerequisites](http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/spot-fleet-requests.html#spot-fleet-prerequisites)

**Logging and troubleshooting**

To increase traceability, use the TRACE log level in the LogLevel parameter in the awsprov_config.json file. This log level prints the entry of the method with the value of the parameters and the exit of the method with the return value (if exists).
The following troubleshooting messages are created when the log level is configured as DEBUG. For troubleshooting purposes, every state change on a Spot instance request is logged with a predefined format:

Spot Fleet Request ID – Spot Instance Request Id- Spot Instance Machine ID: State update message

**Limitations and known issues**

- The Spot Instance Termination Notice is not accurate if the system clock is not synchronized between the master host and the compute host. System clock synchronization is required for reclaim to work.
  

- If a request remains pending for 60 minutes, resource connector assumes that the request is lost. The request is ignored and LSF recalculates the demand. In AWS Spot instances, the request remains pending and is not closed.

- LSF checks periodically for any hosts that are planned to be reclaimed and requeues the jobs within the 2 minute termination notice. However, it's possible that AWS might not honor the 2 minute termination notice, and machines are terminated without a termination notice. For more information, see: [http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/spot-interruptions.html#spot-instance-termination-notices](http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/spot-interruptions.html#spot-instance-termination-notices)

**Configuring AWS Spot instances**

Configure LSF to make Spot instance requests.

1. Configure the pricing resource in the lsf.shared file:

   ```
   Begin Resource
   RESOURCENAME TYPE INTERVAL INCREASING DESCRIPTION
   ...
   pricing String () () (Pricing option: spot/ondemand)
   ...
   End Resource
   ```

2. Configure a Spot instance template in the awsprov_templates.json file.

   The following parameters enable Spot instance requests:

   **vmType**
   Specify a machine type of the AWS instance you want to create. The vmType must support Spot instances.

   Use commas to separate multiple machine types. For example:
   "vmType": "c4.large, m4.large"


   **fleetRole**
   For Spot Instance templates. Specify the role that grants the permission to bid on, launch, and terminate spot fleet instances on behalf of the user.

**spotPrice**

For Spot instance templates. Specify the bid price for the instance. Set a suitable value (usually the On-Demand price) to make sure that the Spot price is equal to or above the market Spot price. The Spot instance is launched when the Spot price of the instance is below the bid specified in the **spotPrice** attribute.

For more information about Spot pricing, see [https://aws.amazon.com/ec2/spot/bid-advisor/](https://aws.amazon.com/ec2/spot/bid-advisor/)

**allocationStrategy**

For Spot instance templates. The allocation strategy for your Spot fleet determines how it fulfills your Spot fleet request from the possible Spot instance pools that are represented by its launch specifications. You can specify the following allocation strategies in your Spot fleet request:

- **lowestPrice**
  - The Spot instances come from the pool with the lowest price. This is the default strategy.

- **diversified**
  - The Spot instances are distributed across all pools.

3. Optional: In the `awsprov_config.json` file, configure the `AWS_SPOT_TERMINATE_ON_RECLAIM` parameter. The `AWS_SPOT_TERMINATE_ON_RECLAIM` parameter processes requests for terminating Amazon EC2 Spot instances that are planned to be reclaimed by AWS.

   If set to `true`, the AWS plugin sends an instance termination request to AWS when notified by LSF that the reclaimed Spot instance has been closed and the affected jobs are requeued.

   Valid values are `true` and `false`. The default value is `false`.

The following template fragment, shows an example configuration for Spot instances:

```json
{
   "Templates": [
   {
      "templateId": "SpotTemplate",
      "attributes": {
         "ncpus": ["Numeric", "4"],
         "awshost": ["Boolean", "1"]
      },
      "imageId": "ami-27ai",
      "vmType": "c4.xlarge, m4.large",
      "subnetId": "subnet-7c0dfb27,subnet-12286475,subnet-cc0248ba",
      "keyName": "LSF-Key",
      "MaxNumber": "10",
      "securityGroupIds": ["sg-7231"]
      "fleetRole": "arn:aws:iam::700071821657:role/EC2-Spot-Fleet-role",
      "spotPrice": "0.1",
      "allocationStrategy": "diversified",
   },
   ]
}
```
Chapter 6. Configuring Microsoft Azure for LSF resource connector

Follow these steps to configure Microsoft Azure to create instances for LSF resource connector to make allocation requests on behalf of LSF. The instances launched from Azure join the LSF cluster. If instances become idle, LSF resource connector terminates them.

- You must have root access to the LSF master host.
- The LSF cluster must have a DNS server.
- You must have correct permissions to update the DNS server.
- You must be able to restart the LSF cluster.
- You must be familiar with and have the ability to perform Microsoft Azure administrative operations.
- The virtual network to be used by Microsoft Azure virtual instances must be configured so that they can communicate with LSF hosts.

LSF resource connector has been tested on the following systems:

- Linux x86 Kernel 2.6, glibc 2.11 RHEL 6.x
- Linux x86 Kernel 3.10, glibc 2.17 RHEL 7.x
- Linux x86 Kernel 3.10, glibc 2.17 CentOS 7.x
- LSF 10.1 Standard Edition

In the following steps, you must perform all operations as the Azure administrator unless otherwise stated.

For full details on installing LSF, see *Installing IBM Spectrum LSF on UNIX and Linux*.

To get the job submitted by a user to run on the instance, the instance must have this user prepared or LSF user mapping configured. For more information about user groups and user account mapping, see "Managing Users and User Groups" and "Between-Host User Account Mapping" in *Administering IBM Spectrum LSF*.

1. Register LSF resource connector, set roles, create authentication file

   Register LSF resource connector as an Azure application, and set roles for LSF resource connector. The LSF resource connector must have access to resources such as images and networks created by the administrator.

   If you have Azure Active Directory administrator permission, and you installed the Azure CLI 2.0, use this link to create an authentication file for the resource connector application:
   
   https://github.com/Azure/azure-sdk-for-java/blob/master/AUTH.md

   If you require accurate access control for LSF resource connector under multiple subscriptions, use the following steps.

   Edit custom role like below to control access rights for LSF resource connector accurately.

   ```
   $ cat custom_role.json
   {
     "Name": "LSF Resource Connector",
     "IsCustom": true,
     "Description": "LSF resource connector for Azure, access/create/delete VM RG storage network",
     "Actions": [
   ```
"Microsoft.Storage/*",
"Microsoft.Network/*",
"Microsoft.Compute/*",
"Microsoft.Authorization/*/read",
"Microsoft.Resources/subscriptions/resourceGroups/*",
"Microsoft.Resources/deployments/*",
"Microsoft.Insights/alertRules/*",
"Microsoft.Insights/diagnosticSettings/*",
"Microsoft.Support/*",
"Microsoft.ResourceHealth/availabilityStatuses/read"
],
"NotActions": [
],
"AssignableScopes": [
  "/subscriptions/1db8ceea-a921-4395-9586-6fc87945f8d7",
  "/subscriptions/5db8ceea-a921-4395-9586-6fc87945f8d9"

}]

Note: List multiple subscriptions in AssignableScopes if you need LSF resource connector to work under multiple subscriptions.

Create a custom role in Azure, then register LSF resource connector as Azure application named "MyAzureApp", and assign custom role "LSF Resource Connector" in two subscriptions.

$ az role definition create --role-definition custom_role.json
$ az ad sp create-for-rbac -o json -n "MyAzureApp" --role "LSF Resource Connector" --scopes "/subscriptions/1db8ceea-a921-4395-9586-6fc87945f8d7" "/subscriptions/5db8ceea-a921-4395-9586-6fc87945f8d9"

2. Create a key pair.

Azure supports public-key cryptography to secure the login information for an instance. A Linux instance has no password; you use a key pair to log in to your instance securely. You specify the ssh public key when you launch your instance, then use the private key when you log in using SSH.

Use the command ssh-keygen to create an SSH key pair.

Provide the content of public key file to LSF to launch Azure instances.
Configure the file path of the public key in the azureprov_templates.json file.

3. Create a security group for LSF.

A network security group (NSG) contains a list of security rules that allow or deny network traffic to resources connected to Azure Virtual Networks (VNet). You need add customized rules to open all LSF listening ports to the security group that are used to launch instances. The ports must match those from the existing LSF cluster. The following are the default port number values:

- **LSF_LIM_PORT=7869** (TCP and UDP)
- **LSF_RES_PORT=6878** (TCP)
- **LSB_SBD_PORT=6882** (TCP)

You can also add master host IP address to accept all traffic from the master host.

**Note:** If you allow the traffic only from the default ports, some NIOS commands might not work, since the ports are configured for them dynamically and are different from the LSF ports.

4. Create a virtual network and subnets that are dedicated to an Azure account.

For more information see the Azure documentation: [https://docs.microsoft.com/en-us/azure/virtual-network/virtual-networks-overview]
A subnet is a range of IP addresses in your virtual network. You can launch Azure resources into a subnet that you select. Use a public subnet for resources that must be connected to the Internet, and a private subnet for resources that won't be connected to the Internet.

You need to specify the name of the subnet in the azureprov_templates.json file which will be used to launch the instance.

5. Build the LSF cloud image.

To create an Azure instance image for an LSF cloud compute host, the cluster administrator must first manually launch an instance and install LSF on that Azure instance.

**Remember:** Choose managed disk when you launch the instance.

For more information, see the Azure documentation: [https://docs.microsoft.com/en-us/azure/virtual-machines/linux/quick-create-portal](https://docs.microsoft.com/en-us/azure/virtual-machines/linux/quick-create-portal)

This is a one-time activity. An image is then created using this instance. Subsequent cloud instances can be dynamically launched using the image just created.

a. After the instance is created, use the ssh command to log in to the instance using the key file that you created in step 2.

b. Copy the LSF packages to Azure instance

   lsf10.1_linux2.6-glibc2.3-x86_64.tar.
   lsf10.1_linux2.6-glibc2.3-x86_64-442293.tar.Z (LSF 10.1SPK2)
   lsf10.1_lsfinstall.tar.Z
   lsf_std_entitlement.dat

c. Install the software.

   ed.x86_64

   If there is no the software installed, you can use command "yum install" to install it.

   #yum install ed

d. Install LSF as a slave host on the Azure instance under the same directory path as the master host. For example, edit the slave.config file with the installation options you need:

   Then run the ./lsfinstall -s -f slave.config command.

   LSF_TOP="/opt/lsf"
   LSF_ADMIN="lsfadmin"
   LSF_TARDIR="/opt/install/"
   LSF_LICENSE="/opt/install/lsf_std_entitlement.dat"
   LSF_SERVER_HOSTS="master.myserver.com"
   LSF_LOCAL_RESOURCES="[resource azurehost]"
   LSF_LIM_PORT=7869
   LSF_GET_CONF=lim

   After installation, make sure that the <LSF_TOP>/conf/lsf.conf file contains the azurehost resource.

   LSF_GET_CONF=lim
   LSF_CONFDIR="/opt/lsf/conf"
   LSF_LIM_PORT=7869
   LSF_SERVER_HOSTS="master.myserver.com"
   LSF_LOCAL_RESOURCES="[resource azurehost]"
   LSF_TOP="/opt/lsf"
   LSF_LOGDIR="/opt/lsf/log"
   LSF_LOG_MASK=LOG_WARNING
   LSF_ENABLE_EGO=N
   LSB_ENABLE_HPC_ALLOCATION=Y
   LSF_EGO_DAEMON_CONTROL=N
**LSF_LIM_PORT**

The port number must be the same as the one defined on the LSF master host.

**LSF_GET_CONF**

Update the LSF configuration to synchronize the cluster configuration with the master host.

**LSF_LOCAL_RESOURCES**

The new resource name azurehost is used by LSF to identify Azure instances. Use the `bhosts -a` command to see instances that are used by LSF.

e. Optional: If required, update the `/etc/hosts` file to add the master host name to the `/etc/hosts` file on the Azure instance, or configure the DNS/NIS client on the instance.

f. Start the LSF daemons on the instance manually and make sure that the instance can join the LSF cluster as a dynamic host.

If not, check the VPN, firewall, or security group settings. Check whether the master host can ping the instance using its private IP address and vice versa. If the master host can ping the instance using its IP address but not using the host name then host name resolution needs to be configured.

g. Shut down the daemons and log out from the instance.

h. Log on Azure instance, deprovision and shutdown instance.

```
sudo waagent -deprovision+user -force && halt
```

i. Capture the image.

Run the following Azure commands.

```
az vm deallocate -g "resource group name" -n "instance name"
az vm generalize -g "resource group name" -n "instance name"
ap image create -n "image name" -g "resource group name" --os-type Linux --source "instance name"
```

The name of the image (imageId) is required in the `azureprov_templates.json` file for LSF to decide when borrowing happens, and when instances are launched from which image.

For more information, see the Azure documentation: [https://docs.microsoft.com/en-us/azure/virtual-machines/linux/capture-image](https://docs.microsoft.com/en-us/azure/virtual-machines/linux/capture-image)

**Related reference:**

"azureprov_templates.json" on page 87

---

**Configuring LSF resource connector for Microsoft Azure**

Specify LSF resource connector configuration to enable Microsoft Azure as a resource provider.

1. Update the Azure resource connector configuration.

   Change the parameters in the `azureprov_config.json` file to enable the resource connector to connect to Azure.

2. Create templates. Create at least one template in the `azureprov_templates.json` file.

   Make sure that your template accurately defines at least the following attributes:

   - `imageId`
   - `vmType`

   The following template is a minimal example for hosts with 4 CPUs:
3. Optional: Update the LSF_TOP parameter in the user_data.sh script to run the hostsetup to start LSF daemons on the instance. This script runs during the Azure instance startup.

Edit the user_data.sh file in the <LSF_TOP>/<LSF_VERSION>/resource_connector/azure-scripts/ directory. For example, specify the directory where LSF is installed on the Azure machine image:

```bash
#!/bin/bash
LSF_TOP=/opt/lsf
source $LSF_TOP/conf/profile.lsf
lsadmin limstartup
lsadmin resstartup
badmin hstartup
```

**Note:** You can add any extra code that needs to run when the Azure instance launches. Upload that script to blob or other URI that can be accessible.
Using the IBM Spectrum LSF Resource Connector
Chapter 7. Configuring IBM Bluemix for LSF resource connector

Follow these steps to configure the IBM Bluemix (formerly SoftLayer) service to enable LSF resource connector to make allocation requests to borrow virtual compute hosts on behalf of LSF. When LSF workload demand exceeds cluster capacity, resource connector generates requests for additional hosts from IBM Bluemix and dispatches jobs to dynamic hosts that join the LSF cluster. When the demand reduces, and hosts become idle, resource connector shuts down slave LSF daemons and cancels allocated IBM Bluemix virtual servers.

The following steps assume that you already have set up an LSF cluster and meet the requirements listed below. They also assume that you have an existing IBM Bluemix (SoftLayer) account.

- You must have an existing LSF cluster set up and have root access to the LSF master host.
- The LSF cluster must have a DNS server.
- You must have correct permissions to update the DNS server.
- You must be able to restart the LSF cluster.
- You must be familiar with the IBM Bluemix console, and have the ability to perform IBM Bluemix administrative operations.
- You must have sufficient permissions to create and configure a custom virtual server image and, optionally, post-provisioning scripts, and order virtual servers on IBM Bluemix.
- The VLAN to be used by IBM Bluemix virtual servers must be configured so that they can communicate with the LSF master host.

LSF resource connector has been tested on the following systems:

- Linux x86 Kernel 3.10, glibc 2.17 CentOS 7.x
- Java Runtime Environment 1.8
- LSF 10.1.0.2 Standard Edition or later

It is expected to work on Linux x86 distributions using kernel 2.6 and higher.

In the following steps, you must perform all operations as the IBM Bluemix administrator unless otherwise stated.

To get the job submitted by a user to run on the instance, the instance must have this user prepared or LSF user mapping configured. For more information about user groups and user account mapping, see "Managing Users and User Groups" and "Between-Host User Account Mapping" in Administering IBM Spectrum LSF.

1. Manually order a SoftLayer computing instance with a standard OS image.
   a. Log in to your account from the SoftLayer Customer Portal [https://control.softlayer.com/].
   b. Click Devices in the Order section and order a virtual server. Select the desired configuration for this virtual server.
   c. In Advanced System Configuration, choose a Host and Domain Name.
   d. Submit the order.
You receive an email notification indicating that the virtual server is currently in the provisioning process. Standard delivery times are 20 minutes or less.

e. Use **Check Devices > Device List** in the SoftLayer portal to see whether the cloud server is provisioned.

After the server is available, you can customize this compute instance.

2. Customize the compute instance by installing IBM Spectrum LSF on it.

To customize the virtual server instance, use SSH (with the public IP as root user). Obtain the root password for the device from the Bluemix portal by clicking the device name and selecting the Passwords tab in the device details.

   a. Use the `scp` command to copy the IBM Spectrum LSF installation packages to the Softlayer instance.

   ```
   lsf10.1_lnx310-1ib217-x86_64.tar.Z
   lsf10.1_lsfinstall_linux_x86_64.tar.Z
   lsf_std_entitlement.dat
   ```

   Use the binary package appropriate for your virtual server OS version.

   b. Log on as root.

   ```
   ssh 169.55.162.62 -l root
   ```

   c. Configure DNS.

   Configure the DNS settings to make sure that the virtual server can look up the host name and IP address of the LSF master host.

   d. Edit the `slave.config` file to configure the LSF installation.

   Extract the `lsf10.1_lsfinstall_linux_x86_64.tar.Z` file into a working directory.

   Typical contents of the `slave.config` file are shown below (the values shown are examples. Use values appropriate for your environment):

   ```
   LSF_TOP="/home/bmuser/lsf"
   LSF_ADMINS="bmuser"
   LSF_TARDIR="/home/bmuser/lsf/
   LSF_ENTITLEMENT_FILE="/home/bmuser/lsf/lsf_std_entitlement.dat"
   LSF_SERVER_HOSTS="master.myserver.com"
   LSF_LOCAL_RESOURCES="[resource softlayercomp] [resource define_ncpus_threads]"
   LSF_LIM_PORT="7869"
   ```

   e. Install LSF on the slave host.

   ```
   ./lsfinstall -s -f slave.config
   ```

   For full details on installing LSF, see *Installing IBM Spectrum LSF on UNIX and Linux*.

   f. Start the LSF daemons on the virtual server manually and make sure that it can join the LSF cluster of the configured master host as a dynamic host.

3. Create the custom image template.

   By saving the previously customized virtual server image as a template you are able to provision a preconfigured LSF slave host at any time.

   Go to **Device Details** for the host at **Devices > Device List > Device Details > Actions**, choose an **Image Name** (for example, `LSFComputeImage`), and click **Create Image Template**.

   When image creation is successful, go to **Devices > Manage > Images** to view the image you created. Remember the **Image Name** for resource connector configuration.

4. Create a provisioning script for the LSF compute node virtual server.

   IBM Bluemix can download the provisioning script from an HTTPS server and run it automatically when the virtual server is started and all its virtual devices...
are connected. The provisioning script is required to configure the LSF compute node to join the correct cluster and present correct shared resources. You can host your provisioning script on another virtual server in IBM Bluemix and configure it to use only the IBM Bluemix internal network for increased security. You can also host your script on any HTTPS server available on the public network.

An example provisioning script is shown below. It reads from the getUserMetadata API to get the configuration variables set by the resource connector during provisioning.

```bash
#!/bin/bash
logfile=/var/log/postprovisionscripts.log
echo START `date +%Y-%m-%d %H:%M:%S` >> $logfile

STARTTIME=`date +%s`
TIMEOUT=60
USERDATA=`curl -s $URL` 2>>$logfile

# while [[ "$USERDATA" == [Nn]"o user data"] ] && [[ `expr $NOWTIME - $STARTTIME` -lt $TIMEOUT ]]; do
  sleep 5
  NOWTIME=`date +%s`
  USERDATA=`curl -s $URL` 2>>$logfile
done

# check if we got user data eventually
if [[ "$USERDATA" == [Nn]"o user data"] ]; then
  # user data is expected to be a semicolon-separated key=value list
  # like environment variables; split them into an array
  #IFS="\";\" read -ra ARR <<<"$USERDATA"
  for VAR in ${ARR[@]}; do
    eval "export $VAR"
  done
else
  echo "$USERDATA: $USERDATA" >>$logfile
  echo EXIT AT `date +%Y-%m-%d %H:%M:%S` >>$logfile
  exit -1
fi

echo "CURRENT ENVIRONMENT:" >>$logfile
env >> $logfile

#Set the correct path for LSF_TOP, where LSF is installed on the VM host
LSF_TOP=/opt/lsf
LSF_CONF_FILE=$LSF_TOP/conf/lsf.conf
source $LSF_TOP/conf/profile.lsf

#Do not remove this part of the script to support rc_account resource for SoftLayer
#You can similarly set additional local resources if needed
if [ -n "$(rc_account)" ]; then
  sed -i 's/\{"LSF_LOCAL_RESOURCES=.*\}"/\{"rc_account\}"/g' $LSF_CONF_FILE
  echo "update LSF_LOCAL_RESOURCES lsf.conf successfully, add [resourceemap ${rc_account}]" >>$logfile
fi

#If there is no DNS server to resolve host names and IPs between master host and VMs,
#then uncomment the following part and set the correct master LSF host name and IP address
#master_host='hostm.example.com'
#master_host_ip='10.115.206.151'
#echo $(master_host_ip) > /etc/hosts
#echo $(master_host) > $LSF_ENVDIR/hostregsetup
#lsreghost -s $LSF_ENVDIR/hostregsetup

#Start LSF Daemons in dynamic VM host.
lshost limstartup
sleep 5
lsadmin ressstartup
```

Chapter 7. Configuring IBM Bluemix for LSF resource connector 47
```bash
badmin hstartup

echo END AT `date +%Y-%m-%d %H:%M:%S` >> $logfile
```

Related reference:

[“softlayer_templates.json” on page 91](#)

---

## Configuring LSF resource connector for IBM Bluemix

Specify LSF resource connector configuration to enable IBM Bluemix as a resource provider.

1. Configure the IBM Bluemix resource connector provider plugin.

   Change the parameters in the `softlayerprov_config.json` file to enable the resource connector to connect to IBM Bluemix.

   The IBM Bluemix configuration file is located in `<LSF_TOP>/conf/resource_connector/softlayer/conf/softlayerprov_config.json`. You must set the `SOFTLAYER_CREDENTIAL_FILE` property to point to a file that contains valid IBM Bluemix API credentials.

   A sample configuration file is shown below.

   ```json
   {
     "SOFTLAYER_CREDENTIAL_FILE": "/opt/lsf/conf/resource_connector/softlayer/conf/credentials",
     "SOFTLAYER_DOMAIN_NAME": "hostfactory.com"
   }
   ```

   The credentials file is typically stored in the same plugin configuration directory, but can be anywhere in the file system. Its contents are two key-value pairs setting IBM Bluemix API username and secret key, for example:

   ```
   softlayer_access_user_name = <your SoftLayer username>
   softlayer_secret_api_key = XXXXXXX
   ```

   You can find the user's API key in the IBM Bluemix console. Select **Account > Users** from the menu and click the **View** link in the **API Key** column.

   At a minimum, the IBM Bluemix user whose credentials are configured for the plugin requires permissions to order and cancel virtual servers and access the necessary network and storage resources.

2. Configure the IBM Bluemix image templates. Create at least one template in the `softlayerprov_templates.json` file.

   Attributes `ncpus` and `softlayercomp` are required. The following template is a minimal example for hosts with 4 CPUs:

   ```json
   {
     "templates": [
       {
         "templateId": "Template-1",
         "maxNumber": 10,
         "attributes": {
           "type": ["String", "X86_64"],
           "ncpus": ["Numeric", "4"],
           "ncores": ["Numeric", "1"],
           "nram": ["Numeric", "8192"],
           "softlayercomp": ["Boolean", "1"],
           "customattr": ["String", "somedata"]
         },
         "imageId": "lsfslavehosts",
         "datacenter": "tor01",
         "vlanNumber": "882",
         "useHourlyPricing": true,
         "localDiskFlag": false,
         "privateNetworkOnlyFlag": false,
         "dedicatedAccountHostOnlyFlag": false,
       }
     ]
   }
   ```

---

Using the IBM Spectrum LSF Resource Connector
Chapter 7. Configuring IBM Bluemix for LSF resource connector
Chapter 8. Submit jobs to LSF resource connector

Check the resource connector status and submit jobs that borrow resources from a resource provider.

Check the resource connector status

On the LSF master host, verify that the **ebrokerd** daemon process is running after the resource connector is enabled.

```bash
# ps -ef | grep ebrokerd
```

Submitting jobs to borrow resources through EGO

Use the **bsub** command to submit jobs that require hosts that are borrowed from IBM Spectrum Conductor with Spark and IBM Spectrum Symphony through EGO. Use the **bhosts** to monitor borrowed hosts. Use the **bhosts** command to monitor host status.

In the examples, conductor01 is a sample compute host in EGO.

1. Use the **bsub** command to submit jobs that require hosts that are borrowed from EGO.
   
The following **bsub** command with no options submits a job that triggers a borrow demand when the LSF cluster has no available resources:
   ```bash
   bsub myjob
   ```
   
   You can also use the conductorhost resource in a **select[]** resource requirement string. Because the conductorhost resource is defined in a template as a Boolean attribute, it triggers a borrow demand:
   ```bash
   bsub -R "select[conductorhost]" myjob
   ```
   
   If a host in the resource group **rg_shared** has no EGO running jobs, EGO lends the host to LSF.

2. Submit batch applications to reclaim a host.

   To submit batch applications so that EGO can reclaim the borrowed host when it requires the lent resources to satisfy its workload demand.

   For more information about submitting batch applications to a Spark instance group, see [www.ibm.com/support/knowledgecenter/SSVH2B_1.1.0/managing_applications/applications_submit.dita](www.ibm.com/support/knowledgecenter/SSVH2B_1.1.0/managing_applications/applications_submit.dita).

3. Use the **bjobs** command to monitor jobs. Jobs running on borrowed resources are requeued.

4. Use the **bhosts** command to monitor borrowed hosts. Host status becomes **ok** when it joins the LSF cluster as a dynamic host.

   ```
<table>
<thead>
<tr>
<th>HOST_NAME</th>
<th>STATUS</th>
<th>JL/U</th>
<th>MAX</th>
<th>NJOBS</th>
<th>RUN</th>
<th>SSUSP</th>
<th>USUSP</th>
<th>RSV</th>
</tr>
</thead>
<tbody>
<tr>
<td>lsfmaster</td>
<td>ok</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>conductor01</td>
<td>ok</td>
<td>-</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
   ```

5. Use the **bhosts** command to monitor host status.

   Use the -a option, which shows all hosts, including hosts that are reclaimed by IBM Spectrum Conductor with Spark.

   During the reclaim grace period, the status of the **conductor01** host changes to **closed** and LSF sends the requeue signal to any jobs that are running on the host.
After the grace period expires or the host becomes idle, IBM Spectrum
Conductor with Spark reclaims the host. LSF daemons on conductor01 are shut
down and its host status becomes unavail.

How LSF returns hosts to EGO

The workflow for returning hosts to EGO occurs when an EGO reclaim request is
triggered.

1. An EGO application requests an allocation from a host in the rg_shared
resource group. A reclaim request is triggered on the host with a grace period.

2. LSF resource connector detects the reclaim request and notifies LSF of the time
that the host needs to be returned.

3. LSF closes the reclaimed host and sends a requeue signal to the jobs on the
host at a configurable time before the requested return time.

4. After the host is drained of jobs, LSF notifies resource connector.

5. Resource connector stops the LSF daemons on the host and deallocates the host
from EGO.

6. EGO detects that the host is now available, and passes the host to the
requesting application.

The following example shows how LSF returns hosts that are reclaimed by EGO:

1. An EGO application makes an allocation request for a slot. The EGO
application is entitled to one slot on host ego01.

2. The `bhosts -w` command shows that host ego01 goes to closed_RC status. The
job is still running.

3. After a while the job is requeued

4. Eventually, host ego01 goes to unavail status and is no longer shown in the
`bhosts` command.

Using the IBM Spectrum LSF Resource Connector
Submitting jobs to launch instances from OpenStack

Use the `bsub` command to submit jobs that require instances that are launched from OpenStack as the resource provider. Use the `bhosts` command to monitor borrowed hosts. Use the `bhosts` command to monitor host status.

In this task, host-10-110-135-193 is a sample instance from OpenStack.

1. Use the `bsub` command to submit jobs that require instances that are launched from OpenStack as the resource provider. The following `bsub` command with no options submits a job that triggers a launch demand when no resources are available in the LSF cluster:

```
bsub myjob
```

You also can use the openstackhost resource in a `select[]` resource requirement string. Because the openstackhost resource is defined in a template as a Boolean attribute, it triggers a launch demand:

```
bsub -R "select[openstackhost]" myjob
```

2. Use the `bhosts` command to monitor instances. The status of the instances becomes `ok` when they join the LSF cluster as dynamic hosts. Verify that the job is running on host-10-110-135-193:

```
bhosts -a
HOST_NAME STATUS JL/U MAX NJOBS RUN SSUSP USUSP RSV
lsfmaster ok - 1 0 0 0 0 0
host-10-110-135-193 ok - 1 1 1 0 0 0
```

3. Use the `bhosts` command to monitor the status of the instances. Run `bhosts` with `-a` option, which shows all hosts, including terminated instances. If no jobs start running on an instance from OpenStack in the number of minutes specified by the `LSF_EXTERNAL_HOST_IDLE_TIME` parameter, the instance is relinquished and its host status changes to `closed_RC`. When the instance is terminated, the host status becomes `unavail`.

You cannot use the `badmin hopen` command to open a borrowed host in `closed_RC` status.

```
bhosts -a
HOST_NAME STATUS JL/U MAX NJOBS RUN SSUSP USUSP RSV
lsfmaster ok - 1 0 0 0 0 0
host-10-110-135-193 unavail - 1 0 0 0 0 0
```

4. (Optional) Use external job submission and execution controls. Use the job submission and execution controls feature to use external, site-specific executable files to validate, modify, and reject jobs, transfer data, and modify the job execution environment. To control job submissions such as permission checks before instances are launched from OpenStack, you can set up an external submission (`esub`) script. For more information, see “External Job Submission and Execution Controls” in Administering IBM Spectrum LSF.

How LSF returns hosts to OpenStack

OpenStack does not actively reclaim a host like EGO does. An allocated VM is passively returned to OpenStack.

- It is idle for the time (configured by the `LSB_RC_EXTERNAL_HOST_IDLE_TIME` parameter in the `lsf.conf` file).
- A time-to-live period expires (configured by the `LSB_RC_EXTERNAL_HOST_MAX_TTL` parameter in `lsf.conf` file).
- If host factory notifies LSF that a host is ready to use, but the host does not join the cluster for 10 minutes. This value is hardcoded.
If the `billingPeriod` parameter is configured in the `hostProviders.json` file, the `LSB_RC_EXTERNAL_HOST_IDLE_TIME` value defined in the `lsf.conf` file is ignored. Resource connector uses the time the instance was launched to return the machine to the provider.

For example, if a host was launched at 10:00 AM, and `billingPeriod` is 60 minutes, and the host became idle at 10:55, it is returned before the next billing cycle that starts at 11 AM.

If set to 0 or not defined, resource connector uses the value defined in the parameter `LSB_RC_EXTERNAL_HOST_IDLE_TIME` in the `lsf.conf` file.

The default billing period is 0.

If the host was previously in the cluster, LSF closes it (`closed_RC` status) and waits for any running jobs on the host to complete. After the host is drained of jobs, LSF notifies the resource connector. The resource connector terminates the OpenStack VM, which deallocates the host.

You cannot use the `badmin hopen` command to open a borrowed host in `closed_RC` status.

---

**Submitting jobs to launch instances from Amazon Web Services**

Use the `bsub` command to submit jobs that require instances that are launched from AWS as the resource provider. Use the `bhosts` to monitor borrowed hosts. Use the `bhosts` command to monitor host status.

In this task, `ip-10-11-13-19` is a sample instance from AWS.

An AWS allocation request happens this way:
1. Resource connector calls a command that makes a machine instance launch request to AWS.
2. After the launch command returns successfully, resource connector notifies LSF that it can use the host after it joins the cluster.
3. When the instance starts, LSF daemons start. When the host joins the cluster, the job is dispatched to the host.
4. Use the `bsub` command to submit jobs that require instances that are launched from AWS as the resource provider. The following `bsub` command with no options submits a job that triggers a launch demand when no available resources are in the LSF cluster:

   ```
   bsub myjob
   ```

   You also can use the `awshost` resource in a `select[]` resource requirement string. Because the `awshost` resource is defined in a template as a Boolean attribute, it triggers a launch demand:

   ```
   bsub -R "select[awshost]" myjob
   ```

2. Use the `bhosts` command to monitor instances. The status of the instances becomes `ok` when they join the LSF cluster as dynamic hosts. Verify that the job is running on `ip-10-11-13-19`:

   ```
   bhosts -a
<table>
<thead>
<tr>
<th>HOST_NAME</th>
<th>STATUS</th>
<th>JL/U</th>
<th>MAX NJOBS</th>
<th>RUN</th>
<th>SSUSP</th>
<th>USUSP</th>
<th>RSV</th>
</tr>
</thead>
<tbody>
<tr>
<td>lsfmaster</td>
<td>ok</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ip-10-11-13-19</td>
<td>ok</td>
<td>-</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
   ```

   You can use the `bhosts` command to monitor instances. The status of the instances becomes `ok` when they join the LSF cluster as dynamic hosts. Verify that the job is running on `ip-10-11-13-19`:

---

Using the IBM Spectrum LSF Resource Connector
3. Use the `bhosts` command to monitor the status of the instances.

Run `bhosts` with `-a` option, which shows all hosts, including terminated instances.

If an instance from AWS has no running jobs on it in the number of minutes specified by the `LSF_EXTERNAL_HOST_IDLE_TIME` parameter, it is relinquished and its host status changes to `closed_RC`, then `unavail` when the instance is terminated.

You cannot use the `badmin hopen` command to open a borrowed host in `closed_RC` status.

```
3. Use the `bhosts` command to monitor the status of the instances.
    Run `bhosts` with `-a` option, which shows all hosts, including terminated instances.

    If an instance from AWS has no running jobs on it in the number of minutes specified by the `LSF_EXTERNAL_HOST_IDLE_TIME` parameter, it is relinquished and its host status changes to `closed_RC`, then `unavail` when the instance is terminated.

    You cannot use the `badmin hopen` command to open a borrowed host in `closed_RC` status.
```

```
<table>
<thead>
<tr>
<th>HOST_NAME</th>
<th>STATUS</th>
<th>J/L/U</th>
<th>MAX NJOBS</th>
<th>RUN</th>
<th>SSUSP</th>
<th>USUSP</th>
<th>RSV</th>
</tr>
</thead>
<tbody>
<tr>
<td>lsfmaster</td>
<td>ok</td>
<td>-</td>
<td>1 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ip-10-11-13-19</td>
<td>unavail</td>
<td>-</td>
<td>1 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
```

If an instance is in the cluster more than the number of minutes specified by the `LSB_RC_EXTERNAL_HOST_MAX_TTL` parameter, it is closed (`closed` status) and any running jobs on the instance are allowed to run to completion. After the instance is idle, it is terminated and its status becomes `unavail`.

```
3. Use the `bhosts` command to monitor the status of the instances.
    Run `bhosts` with `-a` option, which shows all hosts, including terminated instances.

    If an instance is in the cluster more than the number of minutes specified by the `LSB_RC_EXTERNAL_HOST_MAX_TTL` parameter, it is closed (`closed` status) and any running jobs on the instance are allowed to run to completion. After the instance is idle, it is terminated and its status becomes `unavail`.
```

```
<table>
<thead>
<tr>
<th>HOST_NAME</th>
<th>STATUS</th>
<th>J/L/U</th>
<th>MAX NJOBS</th>
<th>RUN</th>
<th>SSUSP</th>
<th>USUSP</th>
<th>RSV</th>
</tr>
</thead>
<tbody>
<tr>
<td>lsfmaster</td>
<td>ok</td>
<td>-</td>
<td>1 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ip-10-11-13-19</td>
<td>closed</td>
<td>-</td>
<td>1 1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
```

4. Optional: Use external job submission and execution controls.

Use the job submission and execution controls feature to use external, site-specific executable files to validate, modify, and reject jobs, transfer data, and modify the job execution environment. To control job submissions, such as permission checks before instances are launched from AWS, you can set up an external submission (`esub`) script.

For more information, see "External Job Submission and Execution Controls" in `Administering IBM Spectrum LSF`.

AWS can reclaim EC2 Spot instances. Amazon EC2 reclaims a spot instance when the Spot price is greater than the bid price placed in the request.

- Every 30 seconds (the `LSB_RC_QUERY_INTERVAL`, configured in the `lsf.conf` file), a request is sent to AWS to check if any machine is eligible to be reclaimed by AWS. AWS provides a 2 minute termination notice.
- If an instance is marked to be terminated or was already terminated, resource connector sends a relinquish machine request to LSF.
  - If the `AWS_SPOT_TERMINATE_ON_RECLAIM=true` parameter is set in the `awsprov_config.json` file, the AWS plug in sends an instance termination request to AWS when notified by LSF that the reclaimed Spot instance has been closed and the affected jobs are requened.
  - If the `AWS_SPOT_TERMINATE_ON_RECLAIM=false` or the parameter is set in the `awsprov_config.json` file, the AWS plug in does not send an instance termination request and allows the instance to be terminated by AWS.

The default is `AWS_SPOT_TERMINATE_ON_RECLAIM=false`.

Specify `AWS_SPOT_TERMINATE_ON_RECLAIM=false` has if you want AWS will reclaim the host. If the host is reclaimed in the middle of an instance hour, you will not be charged for this hour. For more information, see: [http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-spot-instances.html#spot-pricing](http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-spot-instances.html#spot-pricing) The disadvantage is that the host is no longer managed by LSF. It is AWS's responsibility to terminate the host. LSF relies on AWS to perform the termination. If the termination is not done, extra charges can occur.
LSF closes the reclaimed hosts and sends a requeue signal to the jobs on the host at a configurable time before the requested return time. After the hosts are drained of jobs, LSF notifies resource connector that the hosts are closed.

How LSF returns hosts to AWS
Amazon Web Services never reclaims a host actively like EGO does. Borrowed AWS instances are returned passively.

- It is idle for the time (configured by the \texttt{LSB\_EXTERNAL\_HOST\_IDLE\_TIME} parameter in the \texttt{lsf.conf} file).
- A time-to-live period expires (configured by the \texttt{LSB\_EXTERNAL\_HOST\_MAX\_TTL} parameter in \texttt{lsf.conf} file).
- If host factory notifies LSF that a host is ready to use, but the host does not join the cluster for 10 minutes. This value is hardcoded.

If the \texttt{billingPeriod} parameter is configured in the \texttt{hostProviders.json} file, the \texttt{LSB\_EXTERNAL\_HOST\_IDLE\_TIME} value defined in the \texttt{lsf.conf} file is ignored. Resource connector uses the time the instance was launched to return the machine to the provider.

For example, if a host was launched at 10:00 AM, and \texttt{billingPeriod} is 60 minutes, and the host became idle at 10:55, it is returned before the next billing cycle that starts at 11 AM.

If set to 0 or not defined, resource connector uses the value defined in the parameter \texttt{LSB\_EXTERNAL\_HOST\_IDLE\_TIME} in the \texttt{lsf.conf} file.

If the host was previously in the cluster, LSF closes it (\texttt{closed\_RC} status) and waits for any running jobs on the host to complete. After the host is drained of jobs, LSF notifies the resource connector. The resource connector deallocates the host by terminating the instance in AWS.

You cannot use the \texttt{badmin hopen} command to open a borrowed host in \texttt{closed\_RC} status.

Submitting jobs to launch instances from Microsoft Azure
Use the \texttt{bsub} command to submit jobs that require instances that are launched from Microsoft Azure as the resource provider. Use the \texttt{bhosts} to monitor borrowed hosts. Use the \texttt{bhosts} command to monitor host status.

In this task, \texttt{ip-10-11-13-19} is a sample instance from Microsoft Azure.

1. Use the \texttt{bsub} command to submit jobs that require instances that are launched from Microsoft Azure as the resource provider. The following \texttt{bsub} command with no options submits a job that triggers a launch demand when no resources are available in the LSF cluster:

\begin{verbatim}
bsub myjob
\end{verbatim}

You also can use the azurehost resource in a select[] resource requirement string. Because the azurehost resource is defined in a template as a Boolean attribute, it triggers a launch demand:

\begin{verbatim}
bsub -R "select[azurehost]" myjob
\end{verbatim}

2. Use the \texttt{bhosts} command to monitor instances. The status of the instances becomes \texttt{ok} when they join the LSF cluster as dynamic hosts. Verify that the job runs on \texttt{ip-10-11-13-19}:
3. Use the `bhosts` command to monitor the status of the instances. Run `bhosts` with `-a` option, which shows all hosts, including terminated instances. If no jobs start running on an instance from OpenStack in the number of minutes specified by the `LSF_EXTERNAL_HOST_IDLE_TIME` parameter, the instance is relinquished and its host status changes to `closed_RC`. When the instance is terminated, the host status becomes `unavail`.

You cannot use the `badmin hopen` command to open a borrowed host in `closed_RC` status.

4. Optional: Use external job submission and execution controls.

Use the job submission and execution controls feature to use external, site-specific executable files to validate, modify, and reject jobs, transfer data, and modify the job execution environment.

To control job submissions such as permission checks before instances are launched from Azure, you can set up an external submission (esub) script. For more information, see "External Job Submission and Execution Controls" in *Administering IBM Spectrum LSF*.

---

**Submitting jobs to launch instances from IBM Bluemix**

Use the `bsub` command to submit jobs that require instances that are launched from IBM Bluemix as the resource provider. Use the `bhosts` to monitor borrowed hosts. Use the `bhosts` command to monitor host status.

In this task, `slhost01` is a sample virtual server in IBM Bluemix.

1. Use the `bsub` command to submit jobs that require instances that are provisioned by IBM Bluemix as the resource provider. The following `bsub` command with no options submits a job that triggers a launch demand when no resources are available in the LSF cluster:

   ```bash
   bsub myjob
   ```

   You also can use the host resource in a `select[]` resource requirement string. Because the `softlayercomp` resource is defined in a template as a Boolean attribute, it triggers a launch demand:

   ```bash
   bsub -R "select[softlayercomp]" myjob
   ```

   A virtual server order is sent to IBM Bluemix, and after few minutes the new host joins the cluster. Note that the IBM Bluemix provisioning process can take up to 20 minutes.

2. Use the `bhosts` command to monitor the status of the borrowed hosts. Run `bhosts` with `-a` option, which shows all hosts, including terminated instances. LSF can dispatch subsequent jobs to an existing Bluemix virtual server if there is demand and job resource requirements match the host template. The status of the instance becomes `ok` when it joins the LSF cluster as a dynamic host. Verify that the job runs on `slhost01`:
<table>
<thead>
<tr>
<th>HOST_NAME</th>
<th>STATUS</th>
<th>JL/U</th>
<th>MAX NJOBS</th>
<th>RUN</th>
<th>SSUSP</th>
<th>USUSP</th>
<th>RSV</th>
</tr>
</thead>
<tbody>
<tr>
<td>lsfmaster</td>
<td>ok</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>slhost01</td>
<td>ok</td>
<td>-</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

If there is no additional demand for the Bluemix compute host, LSF relinquishes the host and a cancel request is sent to Bluemix. After some time the virtual server will become unavailable.

<table>
<thead>
<tr>
<th>HOST_NAME</th>
<th>STATUS</th>
<th>JL/U</th>
<th>MAX NJOBS</th>
<th>RUN</th>
<th>SSUSP</th>
<th>USUSP</th>
<th>RSV</th>
</tr>
</thead>
<tbody>
<tr>
<td>lsfmaster</td>
<td>ok</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>slhost01</td>
<td>unavail</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Chapter 9. Logging and troubleshooting the LSF resource connector

Log files for the resource connector are located in the log directory that is defined by the LSF_LOGDIR parameter in the lsf.conf file.

**Log files for LSF**

To change the log level or log classes for LSF, update the following parameters in the lsf.conf file:
- LSF_LOG_MASK
- LSB_DEBUG_MBD
- LSB_DEBUG_EBROKERD

For example, the following parameters set the log level to LOG_INFO, and the debugging log class for the mbatchd and ebrokerd daemons to LC2_RC:

```
LSF_LOG_MASK=LOG_INFO
LSB_DEBUG_MBD="LC2_RC"
LSB_DEBUG_EBROKERD="LC2_RC"
```

**Log files for the resource connector**

To change the log level for the resource connector, update the LogLevel parameter in the resource provider configuration files:
- For EGO: `<LSF_TOP>/conf/resource_connector/ego/conf/egoprov_config.json`
- For OpenStack: `<LSF_TOP>/conf/resource_connector/openstack/conf/osprov_config.json`
- For AWS: the `<LSF_TOP>/conf/resource_connector/aws/conf/awsprov_config.json`

For example, set the log level to INFO:
```
{
    "LogLevel": "INFO",
}
```

**Persistent files for the resource connector**

The resource connector saves some state information for synchronization with LSF after failover or restart. This information is saved in persistent files, which are in the `<LSB_SHAREDIR>/<cluster_name>/resource_connector/` directory.
Chapter 10. Reference for LSF resource connector

Reference for configuring LSF resource connector. For detailed information about LSF configuration parameters, see the IBM Spectrum LSF Configuration Reference.

lsb.applications

Configure the operation of LSF resource connector in the lsb.applications file.

**RC_ACCOUNT**

Assigns an account name (tag) to hosts borrowed through LSF resource connector, so that they cannot be used by other user groups, users, or jobs.

**Syntax**

RC_ACCOUNT = account_name

**Description**

When a job is submitted to an application profile with the RC_ACCOUNT parameter specified, hosts borrowed to run the job are tagged with the value of the RC_ACCOUNT parameter. The borrowed host cannot be used by other applications that have a different value for the RC_ACCOUNT parameter (or that don’t have the RC_ACCOUNT parameter defined at all).

After the borrowed host joins the cluster, use the lshosts -s or lshosts -l command to view the value of the RC_ACCOUNT parameter for the host.

**Example**

RC_ACCOUNT=project1

**Default**

No account defined for the application profile

lsb.queues

Configure the operation of LSF resource connector in the lsb.queues file.

**RC_ACCOUNT**

Assigns an account name (tag) to hosts borrowed through LSF resource connector, so that they cannot be used by other user groups, users, or jobs.

**Syntax**

RC_ACCOUNT = account_name

**Description**

When a job is submitted to a queue with the RC_ACCOUNT parameter specified, hosts borrowed to run the job are tagged with the value of the RC_ACCOUNT parameter.
The borrowed host cannot be used by other queues that have a different value for the **RC_ACCOUNT** parameter (or that don’t have the **RC_ACCOUNT** parameter defined).

After the borrowed host joins the cluster, use the `lshosts -s` or `lshosts -l` command to view the value of the **RC_ACCOUNT** parameter for the host.

**Example**

```plaintext
RC_ACCOUNT=project1
```

**Default**

No account defined for the queue

**RC_DEMAND_POLICY**

Defines threshold conditions for the determination of whether demand is triggered to borrow resources through resource connector for all the jobs in a queue. As long as pending jobs at the queue meet at least one threshold condition, LSF expresses the demand to resource connector to trigger borrowing.

**Syntax**

```plaintext
RC_DEMAND_POLICY = THRESHOLD[ [ num_pend_jobs[duration] ] ... ]
```

**Description**

The demand policy defined by the **RC_DEMAND_POLICY** parameter can contain multiple conditions, in an OR relationship. A condition is defined as `[ num_pend_jobs[duration] ]`. The queue has more than the specified number of eligible pending jobs that are expected to run at least the specified duration in minutes. The `num_pend_jobs` option is required, and the duration is optional. The default duration is 0 minutes.

LSF considers eligible pending jobs for the policy. An ineligible pending job (for example, a job dependency is not satisfied yet) keeps pending even thought hosts are available. The policy counts a job for eligibility no matter how many tasks or slots the job requires. Each job element is counted as a job. Pending demand for a resizable job is not counted, though LSF can allocate borrowed resources to the resizable job.

LSF evaluates the policies at each demand calculation cycle, and accumulates duration if the `num_pend_jobs` option is satisfied. The `mbschd` daemon resets the duration of the condition when it restarts or if the condition has not been evaluated in the past 2 minutes. For example, if no pending jobs are in the cluster, for 2 minutes, `mbschd` stops evaluating them.

**Example**

In the following example, LSF calculates demand if the queue has 5 or more pending jobs in past 10 minutes, or 1 or more pending jobs in past 60 minutes, or 100 or more pending jobs.

```plaintext
RC_DEMAND_POLICY = THRESHOLD[ [ 5, 10 ] [1, 60] [100] ]
```

**Default**

Not defined for the queue
RC_HOSTS

Enables LSF resource connector to borrow specific host types from a resource provider.

Syntax

RC_HOSTS=string

RC_HOSTS = none | all | host_type [host_type ...]

Description

The host_type flag is a Boolean resource that is a member of the list of host resources that are defined in the LSB_RC_EXTERNAL_HOST_FLAG parameter in the lsf.conf file.

If the RC_HOSTS parameter is not defined in the queue, its default value is none. Borrowing is disabled for any queue that explicitly defines RC_HOSTS=none, even if the LSB_RC_EXTERNAL_HOST_FLAG parameter is defined in the lsf.conf file.

If the RC_HOSTS parameter is not defined in any queue, borrowing cannot happen for any job.

Note: The HOSTS parameter in the lsb.queues file and the bsub -m option do not apply to hosts that are managed through the resource connector. To specify the resource connector host types that can be used by a queue or an application profile, you must specify the RC_HOSTS parameter in a queue or application profile.

Example

RC_HOSTS=awshost

Default

none - host borrowing from resource providers is disabled, and no borrowed hosts can be used by the queue.

lsf.conf

Enable and configure the operation of LSF resource connector in the lsf.conf file.

LSB_RC_DEFAULT_HOST_TYPE

LSF resource connector default host type.

Syntax

LSB_RC_DEFAULT_HOST_TYPE=string

Description

Specifies the default host type to use for a template if the type attribute is not defined on a template in the template configuration files (egoprov_templates.json, osprov_templates.json, or awsprov_templates.json).

Example

LSB_RC_DEFAULT_HOST_TYPE=X86_64
LSB_RC_EXTERNAL_HOST_FLAG

Setting the **LSB_RC_EXTERNAL_HOST_FLAG** parameter enables the LSF resource connector feature.

**Syntax**

```bash
LSB_RC_EXTERNAL_HOST_FLAG="string ...
```

**Description**

Specify a list of Boolean resource names that identify borrowed hosts or OpenStack and AWS instances. Any hosts or instances that provide a resource from the list are initially closed by LSF at startup. Hosts and instances are only opened when the resource connector informs LSF that the host was successfully allocated or the instance is launched.

Run the `badmin mbdrestart` command for this parameter to take effect.

**Example**

```bash
LSB_RC_EXTERNAL_HOST_FLAG="conductorhost"
```

**Default**

Not defined

---

LSB_RC_EXTERNAL_HOST_IDLE_TIME

For LSF resource connector. If no jobs are running on a resource provider instance for the specified number of minutes, LSF shuts down the instance.

**Syntax**

```bash
LSB_RC_EXTERNAL_HOST_IDLE_TIME=minutes
```

**Description**

If the **LSB_RC_EXTERNAL_HOST_IDLE_TIME** parameter is set to 0, the policy is disabled and the resource provider instance is never shut down for lack of jobs.

**Example**

```bash
LSB_RC_EXTERNAL_HOST_IDLE_TIME=30
```

**Default**

60 minutes

---

LSB_RC_EXTERNAL_HOST_MAX_TTL

For LSF resource connector. Maximum time-to-live for a resource provider instance. If an instance is in the cluster for this number of minutes, LSF closes it and its status goes to `closed_RC`.

---

Using the IBM Spectrum LSF Resource Connector
Syntax

LSB_RC_EXTERNAL_HOST_MAX_TTL=minutes

Description

If the resource provider becomes idle after the specified number of minutes, it is shut down in the resource provider. If the **LSB_RC_EXTERNAL_HOST_MAX_TTL** parameter is set to 0, the policy is disabled. The resource provider instance is never shut down or relinquished when it becomes idle.

You cannot use the **badmin hopen** command to open a borrowed host in closed_RC status.

Example

LSB_RC_EXTERNAL_HOST_MAX_TTL=30

Default

0 minutes (disabled)

**LSB_RC_QUERY_INTERVAL**

For LSF resource connector. The interval in seconds that the resource connector checks host status and asynchronous requests from a resource provider.

Syntax

LSB_RC_QUERY_INTERVAL=seconds

Description

The interval in seconds that the resource connector checks host status and asynchronous request results from IBM Spectrum Conductor with Spark, OpenStack, or AWS.

Run **badmin mbdrerstart** for any change to take effect.

Example

LSB_RC_QUERY_INTERVAL=60

Default

30 seconds

**LSB_RC_REQUEUE_BUFFER**

For LSF resource connector. The number of seconds before the expiration of the reclaim grace period before which LSF starts to send requeue signals to running jobs on reclaimed hosts.

Syntax

LSB_RC_REQUEUE_BUFFER=seconds
**Description**

The minimum value is 1, which means that LSF sends the requeue signal 1 second before the hosts are reclaimed and the daemons are shut down. Use a low value for jobs on reclaimed hosts to run longer.

*Note:* A low value increases the risk that the requeue operation does not complete before the host is reclaimed and the daemons are shut down. If a host is reclaimed before the requeue is complete, the jobs on the host go to UNKNWN status until the host is returned to LSF.

**Example**

```
LSB_RC_REQUEUE_BUFFER=20
```

**Default**

30 seconds

**LSB_RC_UPDATE_INTERVAL**

For LSF resource connector. Configures how often LSF calculates demand for pending jobs and publishes this demand to the `ebrokerd` daemon.

**Syntax**

```
LSB_RC_UPDATE_INTERVAL=seconds
```

**Description**

This parameter updates the demand calculation according to the specified interval instead of calculating demand every scheduler cycle to avoid performance impact.

**Example**

```
LSB_RC_UPDATE_INTERVAL=20
```

**Default**

30 seconds

---

**egoprov_config.json**

The `egoprov_config.json` file manages remote administrative functions that the resource connector must perform on both the IBM Spectrum Conductor with Spark and LSF clusters.

For example, you can configure the `egoprov_config.json` file to start or stop LSF daemons that are running on IBM Spectrum Conductor with Spark shared hosts.

The default location for the file is

```
<LSF_TOP>/conf/resource_connector/ego/conf/egoprov_config.json
```

**Parameters**

The file is a JSON format tuple that contains the following parameter fields:
LogLevel
The log level for the host provider. Use the following log levels:

- INFO (the default level)
- DEBUG
- WARN
- ERROR
- FATAL

EGOConf
Required. Configuration file location for connection settings for the IBM Spectrum Conductor with Spark master host. The default value is egoprov_ego.conf. Change this parameter only when you want to use an existing configuration file.

Username
The user name of the IBM Spectrum Conductor with Spark administrator.

Password
The IBM Spectrum Conductor with Spark administrator password. The password is used by the resource connector to authenticate with IBM Spectrum Conductor with Spark as an administrator. Configure the password as plain text. The password is encrypted and rewritten to this parameter after the first use.

ClientName
The resource connector uses this client name to register to the IBM Spectrum Conductor with Spark EGO service. The default value is ResProvider.

AllocationName
The allocation name that is requested by EGO client. The default value is ResProvider.

StartLSFCmd
Required. The resource connector uses this script to start LSF daemons on the requested shared hosts. Set this parameter to the location of the LSF startup script.

StopLSFCmd
Required. The resource connector uses this script to stop LSF daemons on the requested shared hosts. Set this parameter to the location of the LSF shutdown script.

Umask
Required. The umask value to run commands on the requested shared host. The default value is 0777.

GracePeriod
Required. The number of seconds to wait for LSF to return borrowed hosts before forcefully shutting down LSF daemons.

The default value is 60.

The minimum value for GracePeriod is LSF_RC_QUERY_INTERVAL + LSB_RC_REQUEUE_BUFFER. The maximum value for GracePeriod is consumer's reclaim grace period plus 120 seconds. For example, if the grace period or timeout for a consumer is 30 seconds, the GracePeriod parameter must be less than 150 seconds (120+30=150); otherwise, LSF waits too long to requeue jobs.

You can find consumer's reclaim grace period from the management console: Resources > Consumers > consumer_name > Consumer Properties > Reclaim behavior. For more information about how resources are reclaimed in IBM
Spectrum Conductor with Spark, see [http://www.ibm.com/support/knowledgecenter/SSVH2B_1.1.0/shared_files/resource_reclaim_configuring.dita](http://www.ibm.com/support/knowledgecenter/SSVH2B_1.1.0/shared_files/resource_reclaim_configuring.dita)

**ExecCwd**
Required. The current working directory (cwd) where the StartLSFCmd and StopLSFCmd commands that are run on borrowed hosts. The default value is /tmp.

**EGOSharedHostsResourceGroup**
Required. A list of string pairs that specify IBM Spectrum Conductor with Spark resource groups from which LSF can borrow shared resources, and the associated consumer name that is used to request allocations from that group.

Each pair must define two parameters:

- **Name**
  Required. The Platform Conductor resource group name.

- **Consumer4LSF**
  Required. The corresponding consumer name that has access to the specified resource group.

**Example egoprov_config.json file**

```json
{
  "LogLevel": "INFO",
  "EGOConf": "egoprov_ego.conf",
  "Username": "Admin",
  "Password": "Admin",
  "ClientName": "ResProvider",
  "AllocationName": "LSF_Alloc",
  "StartLSFCmd": "/usr/share/lsf/lsf.start",
  "StopLSFCmd": "/usr/share/lsf/lsf.stop",
  "Umask": "0777",
  "GracePeriod": "60",
  "ExecCwd": "/tmp",
  "EGOSharedHostsResourceGroup": [
    {
      "Name": "ConductorHosts_TmplA",
      "Consumer4LSF": "TmplA_LSFCons"
    }
  ]
}
```

**egoprov_ego.conf**

The `egoprov_ego.conf` file contains connection settings for IBM Spectrum Conductor with Spark.

The default location for the file is

`<LSF_TOP>/conf/resource_connector/ego/conf/egoprov_ego.conf`

**Description**

The resource Connector uses the `egoprov_ego.conf` file to connect to a remote IBM Spectrum Conductor with Spark master host. The parameter values in the `egoprov_ego.conf` file must match the corresponding parameters in the `ego.conf` file in the IBM Spectrum Conductor with Spark cluster.

The following are the minimum required parameters:

- **EGO_MASTER_LIST**
• EGO_LIM_PORT
• EGO_KD_PORT
• EGO_PEM_PORT
• RC_EGO_VERSION

EGO_MASTER_LIST
  Required. Platform Conductor master host list.

EGO_LIM_PORT
  Required. EGO lim daemon port number on the IBM Spectrum Conductor
  with Spark master host. The default value is 7869.

EGO_KD_PORT
  Required. EGO vemkd daemon port number on the IBM Spectrum Conductor
  with Spark master host. The default value is 7870.

EGO_PEM_PORT
  Required. EGO pem daemon port number on the IBM Spectrum Conductor
  with Spark master host. The default value is 7871.

RC_EGO_VERSION
  Required. This parameter refers to the version of the EGO libraries that are
  used by the EGO client. Only EGO version 3.3 is supported. In future, resource
  connector supports multiple versions. This parameter value refers to the
  library path of the EGO version:
  
  $LSF_TOP/$LSF_VERSION/$LM_OS/rc_ego_lib/EGO_VERSION/

  $LSF_TOP/$LSF_VERSION/$LM_OS/rc_ego_lib/EGO_VERSION/lib

Example egoprov_ego.conf file

EGO_MASTER_LIST=EGO_master
EGO_LIM_PORT=7869
EGO_KD_PORT=7870
EGO_PEM_PORT=7871
RC_EGO_VERSION=3.3

egoprov_templates.json

The egoprov_templates.json file defines the mapping between LSF resource
demand requests and IBM Spectrum Conductor with Spark hosts.

The default location for the file is

<LSF_TOP>/conf/resource_connector/ego/conf/egoprov_templates.json

Description

A template represents a set of hosts that share some attributes such as number of
CPUs, the amount of available memory, the installed software stack, operating
system, and other attributes.

LSF requests resources from the resource connector by specifying the number of
instances of a particular template that it requires to satisfy its demand. The
resource connector uses the definitions in this file to map this demand into a set of
allocation requests in the IBM Spectrum Conductor with Spark cluster.

Note: When you define templates, you must make sure that the attribute
definitions that are presented to LSF accurately match the attributes that are
provided by EGO hosts returned by the EGO_select string.
For example, if the attribute definition specifies hosts with ncpus=4, but the actual hosts that are returned by IBM Spectrum Conductor with Spark report ncpus=2, the demand calculation in LSF cannot be accurate.

**Parameters**

The file contains a JSON-defined list called templates. Each template in the list is an object that contains the following parameters:

**Name**
- Required. The unique template name.

**Attributes**
- Required. A list of attributes that represent the hosts in the template from the LSF point of view. LSF attempts to place its pending workload on hosts that match these attributes to calculate how many instances of each template to request.

Each attribute string in the list has the following format:
- "attribute_name": ["attribute_type", "attribute_value"]

**attribute_name**
- An LSF resource name, for example, type or ncpus.

The attribute name must either be a built-in resource (such as type or ncpus), or defined in the Resource section in the lsf.shared file on the LSF master host.

**attribute_type**
- Can be either Boolean, String, or Numeric and must correspond to the corresponding resource definition in the lsf.shared file.

**attribute_value**
- The value of the resource that is provided by hosts. For Boolean resources, use 1 to define the presence of the resource and 0 to define its absence. For Numeric resources, specify a range that uses [min:max].

The following attributes have default values if not defined:
- type - the default value is given by the setting of the LSB_RC_DEFAULT_HOST_TYPE in the lsf.conf file. The default value of LSB_RC_DEFAULT_HOST_TYPE is X86_64.
- ncpus - default value is 1.

**EGO_select**
- Required. An EGO resource requirement string that selects the IBM Spectrum Conductor with Spark hosts that match this template.

When LSF requests some instances of a template, the resource connector attempts to create an EGO allocation for the specified number of hosts that match this string. The allocation is created from the resource groups that are defined in the EGOSharedHostsResourceGroup parameter.

To verify that the resource requirement string that is defined matches the hosts that you expect to match with this template, use the following command on the IBM Spectrum Conductor with Spark master host for each resource group:

`egosh resource list -g '<resource_group_name>' -R '<your_ego_select_string>'`
Example egoprov_templates.json file

```json
{
  "Templates": [ 
    { 
      "Name": "TemplateA",
      "Attributes": {
        "type": ["String", "X86_64"],
        "ncpus": ["Numeric", "4"]
      },
      "egohost": ["Boolean", "1"]
    },
    "EGO_select": "select(type=X86_64 & ncpus==4 & mem > 480)"
  ]
}
```

The example defines a template that is named TemplateA. LSF attempts to place any pending workload on hypothetical hosts of type X86_64 with ncpus=4 and mem>480. If pending workload is successfully placed on a number of hosts, it requests the same number of instances of TemplateA to the resource connector. The resource connector, in turn, attempts to allocate the same number of hosts that match the configured EGO_Select expression in the EGO cluster. If any allocations succeed (even if there are fewer than requested), the connector informs LSF that it can use the allocated hosts.

In this example, the template also defines the egohost resource. Use the resource requirement string select[egohost] in your LSF job submission to make sure that your jobs generate demand for EGO resources.

hostProviders.json

The hostProviders.json file configures which resource providers LSF can use.

The default location for the hostProviders.json file is

<LSF_TOP>/conf/resource_connector/hostProviders.json

The hostProviders.json file contains a JSON list of named providers. For example, for OpenStack, the type is openstackProv.

You can define multiple providers in the list, so LSF can borrow hosts from EGO, OpenStack, and AWS if all three providers are properly configured.

For hosts to operate in the same cluster, all host providers must have the same LSF administrator. The LSF administrator must have access to the directories specified by confPath and scriptPath.

You cannot define more than one pre- or post-provisioning script.

The pre- or post-provisioning script is local to each provider defined in the providers list. If you want all providers to run the same script, you need to specify the same path inside each provider.

Each host provider must have a unique name. If two different host providers use the same name, LSF logs a warning and ignores one of the entries.
Parameters

**providers**
- A list of resource providers.

**name**
- The name of the resource provider.

**type**
- The type of the resource provider.

**confPath**
- Path to the resource provider configuration directory. Full and relative paths are supported.
- The default `confPath` is relative to `LSF_TOP/conf/resource_connector`.

**scriptPath**
- Path to the resource provider script directory. Full and relative paths are supported.
- The default `confPath` is relative to `LSF_TOP/LSF_VERSION/resource_connector`.

**scriptOptions**
- By default, LSF assumes that the master host has direct access to AWS. If your site's security policy requires the connection to AWS to be made through a proxy server, the `scriptOptions` attribute enables LSF to connect to AWS instances through the specified proxy host name or IP address, and proxy server port.
- LSF sets environment variable `SCRIPT_OPTIONS` when launching the scripts.

**preProvPath**
- Optional. Resource connector runs the pre-provisioning script specified with absolute path after the instance is created and started successfully but before it is marked allocated to the LSF cluster.

**postProvPath**
- Optional. Resource connector runs the post-provisioning script specified with absolute path after the instance is terminated successfully but before it is removed from the LSF cluster.

**provTimeOut**
- Optional. This parameter is used to avoid the pre- or post-provisioning program from running for unlimited time. Specify a value in minutes.
- If the program doesn’t complete in the specified time, it is ended and reported as failed. Setting the `provTimeOut` value to 0 disables script timeout.
- The default value is 10 minutes. If the pre- or post-provisioning program doesn’t return after 10 minutes, it ends.

**billingPeriod**
- Ignore the `LSB_RC_EXTERNAL_HOST_IDLE_TIME` value defined in the `lsf.conf` file and use the time the instance was launched to return the machine to the provider.
- For example, if a host was launched at 10:00 AM, and `billingPeriod` is 60 minutes, and the host became idle at 10:55, it is returned before the next billing cycle that starts at 11 AM.
- If set to 0 or not defined, resource connector uses the value defined in the parameter `LSB_RC_EXTERNAL_HOST_IDLE_TIME` in the `lsf.conf` file.
The default billing period is 0.

Example

```json
{
  "providers": [
    {
      "name": "ego",
      "type": "egoProv",
      "confPath": "resource_connector/ego",
      "scriptPath": "resource_connector/ego",
      "billingPeriod": "60",
      "preProvPath": "/usr/share/lsf/scripts/pre_provision_ego.sh",
      "postProvPath": "/usr/share/lsf/scripts/post_provision_ego.sh",
      "provTimeOut": 10
    },
    {
      "name": "openstack",
      "type": "openstackProv",
      "confPath": "resource_connector/openstack",
      "scriptPath": "resource_connector/openstack",
      "billingPeriod": "60",
      "preProvPath": "/usr/share/lsf/scripts/pre_provision.sh",
      "postProvPath": "/usr/share/lsf/scripts/post_provision.sh",
      "provTimeOut": 10
    },
    {
      "name": "aws",
      "type": "awsProv",
      "confPath": "resource_connector/aws",
      "scriptPath": "resource_connector/aws",
      "scriptOptions": "-Dhttps.proxyHost=10.115.206.146 -Dhttps.proxyPort=8888",
      "billingPeriod": "60",
      "preProvPath": "/usr/share/lsf/scripts/pre_provision.sh",
      "postProvPath": "/usr/share/lsf/scripts/post_provision.sh",
      "provTimeOut": 10
    },
    {
      "name": "azure",
      "type": "azureProv",
      "confPath": "/usr/share/lsf_nevis/conf/resource_connector/azure",
      "scriptPath": "/usr/share/lsf_nevis/resource_connector/azure"
    },
    {
      "name": "softlayer",
      "type": "softlayerProv",
      "confPath": "resource_connector/softlayer",
      "scriptPath": "resource_connector/softlayer"
    }
  ]
}
```

**osprov_config.json**

The osprov_config.json file manages remote administrative functions that the resource connector must perform on OpenStack.

For example, you can configure the osprov_config.json file to launch or terminate OpenStack virtual instances.

The default location for the file is

```
<LSF_TOP>/conf/resource_connector/openstack/conf/osprov_config.json
```

**Parameters**
The file is a JSON format tuple that contains the following parameter fields:

**LogLevel**
The log level for the host provider. Use the following log levels:
- INFO (the default level)
- DEBUG
- WARN
- ERROR
- FATAL

**InstancePrefix**
Required. Prefix for the OpenStack instance host names. The default value is host, which means that a VM with IP address xx.yy.zz.ww has host name host-xx-yy-zz-ww.

**OS_USERNAME**
Required. The user name that is used to perform the required OpenStack functions. This user must have administrative privileges on OpenStack.

**OS_PASSWORD**
Required. The OpenStack administrator password. Configure the password as plain text. The password is encrypted and rewritten to this parameter after the first use.

**OS_PROJECT_DOMAIN_ID**
Optional. The OpenStack domain ID associated with an LSF project. If not specified, uses the value of the OS_USER_DOMAIN_ID attribute.

**OS_PROJECT_NAME**
Required. The LSF project name in OpenStack with which instances are created.

**OS_AUTH_URL**
Required. The OpenStack authentication URL. The resource connector uses this URL to authenticate with the OpenStack identity service.

**OS_NETWORK_NAME**
Required. OpenStack network name that is attached to instances. This name is the network that instances use to communicate with the LSF cluster.

**OS_DNS_SERVER**
Optional. The DNS server address for OpenStack instances.

**OS_USER_DOMAIN_ID**
Optional. The OpenStack user domain ID, introduced in the OpenStack Identity API version 3. Uses the value default if not specified.

**OS_KEYPAIR**
Optional. The OpenStack key-pair name that is used to log in to instances with ssh. If you do not specify a key-pair, you are not able to log in to the instances manually.

**OS_SECURITYGROUPS**
Optional. A list of strings that specify OpenStack security groups that are applied to instances. If not specified, OpenStack uses the default group.

The OS_USERNAME, OS_PASSWORD and OS_PROJECT_NAME are LSF user credentials created in Chapter 4, “Configuring OpenStack for LSF resource connector,” on page 21.
Example osprov_config.json file

```json
{
    "LogLevel": "INFO",
    "InstancePrefix": "host",
    "OS_USERNAME": "LSF",
    "OS_PASSWORD": "Letmein123",
    "OS_AUTH_URL": "http://119.81.183.245:5000/v3",
    "OS_USER_DOMAIN_ID": "default",
    "OS_PROJECT_NAME": "LSF",
    "OS_KEYPAIR": "LSFKey",
    "OS_SECURITYGROUPS": [
        { "name": "default" }
    ],
    "OS_NETWORK_NAME": "public",
    "OS_DNS_SERVER": "10.110.135.210"
}
```

osprov_templates.json

The osprov_templates.json file defines the mapping between LSF resource demand requests and OpenStack instances.

The template represents a set of hosts that share some attributes such as the number of CPUs, the amount of available memory, the installed software stack, operating system, and other attributes.

LSF requests resources from the resource connector by specifying the number of instances of a particular template that it requires to satisfy its demand. The resource connector uses the definitions in this file to map this demand into a set of allocation requests in OpenStack.

The default location for the file is

\[LSF\_TOP\]/conf/resource_connector/openstack/conf/osprov_templates.json

Description

LSF requests resources from the resource connector by specifying the number of instances of a particular template that it requires to satisfy its demand. The resource connector uses the definitions in this file to map this demand into a set of allocation requests in OpenStack.

Important: When you define templates, you must make sure that the attribute definitions that are presented to LSF exactly match the definitions that are provided by OpenStack. If, for example, the attribute definition specifies hosts with \texttt{ncpus}=4, but the actual hosts that are returned by OpenStack report \texttt{ncpus}=2, the demand calculation in LSF is not accurate.

Parameters

The file contains a JSON-defined list that is named \texttt{templates}. Each template in the list is an object that contains the following parameters:

Name
- Required. The unique template name.

Attributes
- Required. A list of attributes that represent the hosts in the template from the
LSF point of view. LSF attempts to place its pending workload on hosts that match these attributes to calculate how many instances of each template to request.

Each attribute string in the list has the following format:
"attribute_name": ["attribute_type", "attribute_value"]

**attribute_name**
An LSF resource name, for example, type or ncores.

The attribute name must either be a built-in resource (such as r15s or type), or defined in the Resource section in the lsf.shared file on the LSF master host.

**attribute_type**
Can be either Boolean, String, or Numeric and must correspond to the corresponding resource definition in the lsf.shared file.

**attribute_value**
The value of the resource that is provided by hosts. For Boolean resources, use 1 to define the presence of the resource and 0 to define its absence. For Numeric resources, specify a range that uses [min:max].

The following attributes have default values if they are not defined:

**type**
The default value is given by the setting of the `LSB_RC_DEFAULT_HOST_TYPE` in the lsf.conf file. The default value of `LSB_RC_DEFAULT_HOST_TYPE` is X86_64.

**ncpus**
Default value is 1.

**Image**
Required. The image name that is used to start virtual instances of this template.

**Flavor**
Required. The flavor name that is used to start virtual instances of this template.

**MaxNumber**
Required. The maximum number of available instances of this template. LSF never requests more than this number of instances for this template. Set this parameter to an appropriate value according to the instance quota of the project or the maximum capacity of the OpenStack infrastructure.

**UserScript**
Optional. User-written script that is sent to the instance and run inside the instance at start time. It can be used for various operations, for example package installation.

**UserData**
Optional. A string that represents a list of keys and their values. The string has the following format:
<key1>=<value1>;<key2>=<value2>; ...

Each key is converted to uppercase and exported as an environment variable with the specified value inside the instance (and is accessible by the user script). For example, if the **UserData** parameter is defined as: packages=M,N;logfile=X, the following environment variables are exported inside the instance at start time:

**Using the IBM Spectrum LSF Resource Connector**
PACKAGES=M,N
LOGFILE=X

These variables can be read by the UserScript.

**Example egoprov_templates.json file**

```json
{
  "Templates": [
    {
      "Name": "TemplateA",
      "Attributes": {
        "type": ["String", "X86_64"],
        "ncpus": ["Numeric", "4"],
        "mem": ["Numeric", "480"],
        "maxmem": ["Numeric", "512"],
        "openstackhost": ["Boolean", "1"]
      },
      "Image": "CentOS-6.x-LSF-x86_64",
      "Flavor": "m1.small",
      "MaxNumber": "10",
      "UserData": "packages=gcc,g++;logfile=/var/install.log"
    },
    {
      "UserScript": "scripts/userscript.sh"
    }
  ]
}
```

The example defines a template that is named TemplateA. LSF attempts to place any pending workload on hypothetical hosts of type X86_64 with ncpus=4 and mem>480. If LSF successfully places some of its pending workload on N number of hosts, it requests N instances of TemplateA to the resource connector.

If demand is generated for this template, the connector attempts to start virtual machines that match the configured image and flavor in OpenStack. If resource connector succeeds in running the launch command for any of the instances (even if there are fewer than requested), the connector informs LSF that it can use the allocated hosts.

The package list and log file is read from the environment variables PACKAGES and LOGFILE, the UserScript that is included in the instance is run during startup. The following example user script reads these variables and attempts to install the packages:

```bash
#!/bin/bash
yum install -y $(echo $PACKAGES| tr , " ") >> $LOGFILE 2>&1
```

**policy_config.json**

The policy_config.json file configures custom policies for resource providers for LSF resource connector. The resource policy plug-in reads this file.

The default location for the file is

`<LSF_TOP>/conf/resource_connector/policy_config.json`

The policy_config.json file contains a JSON list of named policies. Each policy contains a name, a consumer, a maximum number of instances that can be launched for the consumer, and maximum number of instances that can be launched in a specified period.
Parameters

UserDefinedScriptPath
Optional. Specify the full path to your own resource provider policy script. Your custom policy script runs after the default plug-in runs with the same input JSON file, and the demand that is calculated by your script is used. Demand that is calculated by the default plug-in is ignored. If the UserDefinedScriptPath is defined and it fails to run, the demand is 0, which means no demand.

The following example defines the path to the script Main.py:
"UserDefinedScriptPath" : "/usr/share/lsf/10.1/scripts/Main.py"

Policies
Optional. A list of policies that apply on the demand calculation. If the policies are not defined, the demand that is calculated by resource connector is used.

Name
Required. The name of the policy. You can define multiple policies in the list. Each policy must have a unique name.

Consumer
Optional. The following consumer attributes are supported:

rcAccount
A list of accounts that can borrow hosts through LSF resource connector. Supported values are all or any valid account name that is defined in the RC_ACCOUNT tag in the lsb.queues file.

templateName
A list of template names. Supported values are all or any valid template name.

provider
A list of resource provider names. Supported values are all or any valid provider name.

For any attribute that is not defined, the default value is all.

If a consumer is not defined, the following attributes apply to all providers, templates, accounts defined in the cluster.

MaxNumber
Optional. The maximum number of instances a user can create/launch for the consumer.

The value of MaxNumber must be less than the value of the LSB_RC_MAX_INSTANCES_PER_TEMPLATE parameter that is defined in the lsf.conf file or the maxNumber parameter in the template configuration file.

StepValue
Optional. The StepValue parameter has two values, which are separated by a colon (:). The step index is the maximum number of instances that can be launched at a time for the defined consumer. The step time controls how fast the cluster grows. The step time specifies how long the plug-in waits before it launches another set of instances that are specified by the step value. If the consumer is not defined, the parameter applies cluster wide.

For example, if step value is defined as 5 and step time is defined as 10 ("StepValue": "5:10") and a request comes in for 20 instances, 5 instances are
launched in the first 10 minutes, 5 more in next 10 minutes until the demand is
met or the maximum number instances that are specified by the `MaxNumber`
parameter are launched.

The default for step index to launch all the instances at the same time.

Default Value for step time is 10 minutes. The default value that is applied
only if a step value is defined but a step time is not defined.

Example

```json
{
    "UserDefinedScriptPath" : "/usr/share/lsf/10.1/scripts/Main.py",
    "Policies": [
        {
            "Name": "Policy1",
            "Consumer": {
                "rcAccount": ["all"],
                "templateName": ["all"],
                "provider": ["all"]
            },
            "MaxNumber": "100",
            "StepValue": "5:10"
        },
        {
            "Name": "Policy2",
            "Consumer": {
                "rcAccount": ["default", "project1"],
                "templateName": ["Template-VM-1"],
                "provider": ["aws"]
            },
            "MaxNumber": "50",
            "StepValue": "10:10"
        }
    ]
}
```

**awsprov_config.json**

The `awsprov_config.json` file contains administrative settings for the resource
connector.

For example, you can configure the `awsprov_config.json` file to start remote AWS
services, such as creating virtual instances.

The default location for the file is

```
<LSF_TOP>/conf/resource_connector/aws/conf/awsprov_config.json
```

**Parameters**

The file is a JSON format tuple that contains the following parameter fields:

**LogLevel**

The log level for the host provider. Use the following log levels:

- TRACE
- INFO (the default level)
- DEBUG
- WARN

Chapter 10. Reference 79
AWS_CREDENTIAL_FILE
Defines the path to the AWS authentication file for using AWS IAM credentials. The file contains the user access key and user access secret key. The resource connector uses the credentials in this file to authenticate LSF users with the AWS identity service. These users must have administrative privileges on AWS to perform required AWS functions (start and stop instances, and so on).

If the LSF master host and the resource connector are deployed in an AWS instance, do not include the AWS_CREDENTIAL_FILE parameter. The AWS API credentials are retrieved from the AWS environment automatically.

AWS_CREDENTIAL_SCRIPT
Defines the path to the AWS credential script for using federated accounts with AWS.

You cannot define both the parameters AWS_CREDENTIAL_FILE and AWS_CREDENTIAL_SCRIPT.

Define AWS_CREDENTIAL_SCRIPT only when federated accounts are used with AWS. When AWS_CREDENTIAL_SCRIPT is defined, the AWS_CREDENTIAL_FILE is ignored.

For example:
AWS_CREDENTIAL_SCRIPT=/shared/dir/generateCredentials.py

When neither AWS_CREDENTIAL_FILE nor AWS_CREDENTIAL_SCRIPT parameters are defined, resource connector attempts to retrieve API credentials from an instance profile defined in , under the assumption that it is running in an AWS EC2 instance.

AWS_KEY_FILE
The path to the AWS key pair file. The key pair name is used to log in to instances with ssh. If the AWS_KEY_FILE parameter is not specified, no key file is defined.

AWS_REGION
AWS region name that is attached to instances and user account.

Specifies the region in which the user and the key file are created and where the instances are provisioned. The key file is specific to the region.

AWS_spot_terminate_on_reclaim
Process requests for terminating Amazon EC2 Spot instances that are planned to be reclaimed by AWS.

If set to true, the AWS plugin sends an instance termination request to AWS when notified by LSF that the reclaimed Spot instance has been closed and the affected jobs are requeued.

Valid values are true and false. The default value is false.

Example awsprov_config.json file
{
   "LogLevel": "INFO",
   "AWS_CREDENTIAL_FILE": "/home/aws/credentials",
   "AWS_REGION": "us-east-1",
}

80 Using the IBM Spectrum LSF Resource Connector
awsprov_templates.json

The awsprov_templates.json file defines the mapping between LSF resource demand requests and AWS instances.

The template represents a set of hosts that share some attributes, such as the number of CPUs, the amount of available memory, the installed software stack, operating system.

LSF requests resources from the resource connector by specifying the number of instances of a particular template that it requires to satisfy its demand. The resource connector uses the definitions in this file to map this demand into a set of allocation requests in AWS.

The default location for the file is <LSF_TOP>/conf/resource_connector/aws/conf/awsprov_templates.json

Description

LSF requests resources from the resource connector by specifying the number of instances of a particular template that it requires to satisfy its demand. The resource connector uses the definitions in this file to map this demand into a set of allocation requests in AWS.

Important: When you define templates, you must make sure that the attribute definitions that are presented to LSF exactly match the attributes that are provided by AWS. If, for example, the attribute definition specifies hosts with ncpus=4, but the actual hosts that are returned by AWS report ncpus=2, the demand calculation in LSF is not accurate.

Parameters

The file contains a JSON-defined list called templates. Each template in the list is an object that contains the following parameters:

**templateId**

The unique template name. The **templateId** cannot contain underscores (_).

**Attributes**

A list of attributes that represent the hosts in the template from the LSF point of view. LSF attempts to place its pending workload on hosts that match these attributes to calculate how many instances of each template to request.

You can define any arbitrary string resource in the lsf.shared file and use that as an attribute in the awsprov_templates.json file. You can then use that attribute in a bsub select string (for example, bsub -R "select[zone == us_east_2a]"). If zone == us_east_2a is selected at job submission, hosts are created from the template that defines the zone attribute to us_east_2a.

To submit a job with a specific template name or a template string attribute, you must define that string resource in the lsf.shared and in the user_data.sh script for that resource to be added to the lsf.conf file on the slave host that is created from the template.

The user_data.sh script is located in the <LSF_TOP>/<LSF_VERSION>/resource_connector/aws/scripts directory.

Each attribute string in the list has the following format:

"attribute_name": ["attribute_type", "attribute_value"]
**attribute_name**

An LSF resource name, for example, type or ncores.

The attribute name must either be a built-in resource (such as r15s or type), or defined in the Resource section in the lsf.shared file on the LSF master host.

**attribute_type**

Can be either Boolean, String, or Numeric and must correspond to the corresponding resource definition in the lsf.shared file.

**attribute_value**

The value of the resource that is provided by hosts. For Boolean resources, use 1 to define the presence of the resource and 0 to define its absence. For Numeric resources, specify a range that uses [min:max].

The following attributes have default values if they are not defined:

**type**

The default value is given by the setting of the `LSB_RC_DEFAULT_HOST_TYPE` in the lsf.conf file. The default value of `LSB_RC_DEFAULT_HOST_TYPE` is `X86_64`.

**ncpus**

Default value is 1.

**imageId**

The ID of the Amazon Machine Image (AMI) that has LSF preinstalled on it. This AMI is used to launch virtual instances.

**MaxNumber**

That maximum number of instances to provide. Set the MaxNumber to an appropriate value according to the instance quota of the LSF project.

**subnetId**

The subnet name (virtual private cloud) used to launch virtual instances. Use the subnet through which the instance can communicate with the LSF cluster.

For AWS Spot instances only, you can specify multiple subnets, separated by commas. For example:

```
"subnetId": "subnet-bc219af5, subnet-ac819ch2"
```

Multiple subnets are not supported for on-demand AWS instances.

**vmType**

The machine type of the AWS instance you want to create. The vmType that is configured in each template must correctly represent the template attributes presented to LSF from AWS.

For AWS Spot instances only, you can specify multiple machine types, separated by commas. For example:

```
"vmType": "c4.large, m4.large"
```

Multiple machine types are not supported for on-demand AWS instances.

**fleetRole**

For Spot Instance templates. Specifies the role that grants the permission to bid on, launch, and terminate spot fleet instances on behalf of the user.
**spotPrice**
For Spot instance templates. Specifies the bid price for the instance. The Spot instance is launched when the Spot price of the instance is below the bid specified in the spotPrice attribute.

The spotPrice attribute is used in determining if the request is a spot request or an on-demand request. If you set the spotPrice attribute with a positive number, the AWS plugin considers this request as a spot request. If the attributes fleetRole or allocationStrategy are defined, but the spotPrice is not defined, the request is considered an on-demand request.

If an on-demand request is initiated using a template with multiple vmType or subnetId values, the request fails.

**allocationStrategy**
For Spot instance templates. The allocation strategy for your Spot fleet determines how it fulfills your Spot fleet request from the possible Spot instance pools that are represented by its launch specifications. You can specify the following allocation strategies in your Spot fleet request:

- **lowestPrice**
  The Spot instances come from the pool with the lowest price. This is the default strategy.

- **diversified**
  The Spot instances are distributed across all pools.

**keyName**
The name of the key-pair file that is used by ssh to log in the launched instance. If you don't specify keyName, you can't connect to the instance unless you choose an AMI that is configured to allow users another way to log in.

If you don't have the right to create a new key-pair, a keyName must be specified to create an instance.

**instanceProfile**
Specifies an AWS IAM instance profile to assign to the requested instance. Jobs running in that instance can use the instance profile credentials to access other AWS resources.

The instance profile can be specified by one of the following methods:

- Short name; for example, MyProfile.
  Valid characters for the instance profile name are uppercase and lowercase alphanumeric characters and any of the following ASCII characters: equal sign (=), comma (,), period (.), at sign (@), minus sign (-).
- AWS Amazon Resource Name (ARN); for example, arn:aws:iam::<account number>:instance-profile/LSFRole.
  The colon character (:) cannot appear in the short name or path. The string arn: at the beginning of the profile reference determines whether the reference is an ARN or a short name. Note: In this context “IAM Role” is essentially equivalent to “Instance Profile”.

**instanceTags**
A string that represents a list of keys and their values. These key-value pairs are used to tag the instance, by using Amazon instance tagging feature. If an instance is launched that uses TemplateA, it is tagged with value of the instanceTags attribute defined in TemplateA.

If instanceTags is not specified, LSF still tags the newly launched instances with the following key-value pair:
InstanceID = <ID of the instance created>

The instanceTags attribute also tags EBS volumes with the same tag as the instance. EBS volumes are persistent block storage volumes used with an EC2 instance. EBS volumes are expensive, so you can use the instance ID that tags the volumes for the accounting purposes.

Note: The tags cannot start with the string aws:. This prefix is reserved for internal AWS tags. AWS gives an error if an instance or EBS volume is tagged with a keyword starting with aws:. Resource connector removes and ignores user-defined tags that start with aws:.

securityGroupIds
A list of strings for AWS security groups that are applied to instances. If you don’t specify securityGroupIds, AWS uses the default group.

placementGroupName
The name of the placement group that the instances are launched to. The group must exist on your Amazon account. Successfully launching the instances into a placement group has the following requirements:

- A placement group can’t span multiple Availability Zones.
- The name that you specify for a placement group must be unique within your AWS account.
- The instance type that is defined in the template must be supported by the placement group created.
- Terminate all the instances in the placement group before the placement group is deleted.

UserData
A string that represents a list of keys and their values. The string has the following format:

<key1>=<value1>;<key2>=<value2>; ... 

key
The key name of the UserData, such as "packages, volume, zone, or templateName.

value
A comma-separated list of UserData values, for example, package1, package2.

Each key is converted to uppercase by the resource connector and exported as an environment variable with the specified value inside the instance (and is accessible by the user script). After UserData is defined, it is divided into keys and values and exported to the instance's environment variables.

For example, if the UserData parameter is defined as packages=M,N;logfile=X, the following environment variables are exported inside the instance at start time:

PACKAGES=M,N
LOGFILE=X

These variables can be read by the user_data.sh script in the instance as the keys PACKAGES and ZONE.

Example awsprov_templates.json file

```json
{
  "Templates":
    [  
```
The example defines a template that is named TemplateA. LSF attempts to place any pending workload on hypothetical hosts of type X86_64 with ncpus=4 and mem>480 MB. If LSF successfully places some of its pending workload on N number of hosts, it requests N instances of TemplateA to the resource connector.

If demand is generated for this template, the connector logic attempts to allocate N hosts with the configured image and vmType (instance type) in AWS. If it succeeds to obtain any instances, even if there are fewer than requested, the resource connector informs LSF that it can use the instances.

In this example, the template also defines the awshost resource. You can make sure that your jobs generate demand for AWS resources by using 'select[awshost]' in your LSF job submission resource requirement strings.

The zone attribute is an example string resource that is defined in the lsf.shared file. If the zone attribute is specified, an instance is created in the specified zone.

The user script scripts/user_data.sh is included in the instance and run during instance startup.

The following script is an example of scripts/user_data.sh in a CentOS 6 image. It reads environment variables and updates the lsf.conf file on the instance to define the new zone attribute for that machine.

```
#!/bin/bash
LSF_TOP=/usr/share/lsf
LSF_CONF_FILE=$LSF_TOP/conf/lsf.conf

# run user script to enable selecting template based on zone
%EXPORT_USER_DATA%

logfile=/tmp/userscript.log
echo > $logfile
if [ -n "$\{zone\}" ]; then
    sed -i "s/\(LSF_LOCAL_RESOURCES=.*\)"/\1[resource \{zone\}] [resourcemap \{zone\}]*zone"/ "$LSF_CONF_FILE"
    echo "update LSF_LOCAL_RESOURCES lsf.conf successfully,
```
add [resource ${zone}] [resourcemap ${zone}*zone]" >> $logfile
else
echo "zone doesn't exist in environment variable" >> $logfile
fi

The following template creates on-demand instances:
{
  "templates": [
    {
      "templateId": "templateA",
      "attributes": {
        "type": ["String", "X86_64"],
        "ncores": ["Numeric", "1"],
        "ncpus": ["Numeric", "1"],
        "nram": ["Numeric", "512"],
        "awshost1": ["Boolean", "1"],
        "zone": ["String", "us_west_2a"],
        "pricing": ["String", "ondemand"]
      },
      "imageId": "ami-8914cbe9",
      "subnetId": "subnet-cc0248ba",
      "vmType": "t2.nano",
      "KeyName": "martin",
      "securityGroupIds": ["sg-b35182ca"],
      "instanceTags": "Name=aws1-vm-1-from-cluster-aws1",
      "userData": "zone=us_west_2a; pricing=ondemand"
    }
  ]
}

The following template creates Spot instances:
{
  "templates": [
    {
      "templateId": "templateB",
      "attributes": {
        "type": ["String", "X86_64"],
        "ncores": ["Numeric", "1"],
        "ncpus": ["Numeric", "1"],
        "nram": ["Numeric", "512"],
        "awshost1": ["Boolean", "1"],
        "zone": ["String", "us_west_2b"],
        "pricing": ["String", "spot"]
      },
      "imageId": "ami-8914cbe9",
      "subnetId": "subnet-cc0248ba",
      "vmType": ["c4.xlarge", "m4.large"],
      "fleetRole": "arn:aws:iam::700071821657:role/EC2-Spot-Fleet-role",
      "securityGroupIds": ["sg-b35182ca"],
      "spotPrice": "0.1",
      "allocationStrategy": "diversified",
      "instanceTags": "Name=aws1-vm-3-spot-aws1",
      "userData": "zone=us_west_2b; pricing=spot"
    }
  ]
}

azureprov_config.json

The azureprov_config.json file manages remote administrative functions that the resource connector must perform on Microsoft Azure.
For example, you can configure the `azureprov_config.json` file to invoke remote Azure services, such as creating virtual instances.

The default location for the file is

```
<LSF_TOP>/conf/resource_connector/azure/conf/azureprov_config.json
```

**Parameters**

The file is a JSON format tuple that contains the following parameter fields:

- **LogLevel**
  - The log level for the host provider. Use the following log levels:
    - INFO (the default level)
    - DEBUG
    - WARN
    - ERROR
    - FATAL

- **AZURE_CREDENTIAL_FILE**
  - The Azure authentication file. The resource connector uses the credentials in this file to authenticate LSF users with the Azure Identity server.

- **AZURE_REGION**
  - Azure location name where instances and resources are placed.

**Example `azureprov_config.json` file**

```
{
  "LogLevel": "INFO",
  "AZURE_CREDENTIAL_FILE": "/home/azure/credentials",
  "AZURE_REGION": "southeastasia"
}
```

**azureprov_templates.json**

The `azureprov_templates.json` file defines the mapping between LSF resource demand requests and Microsoft Azure instances.

The template represents a set of hosts that share some attributes such as the number of CPUs, the amount of available memory, the installed software stack, operating system, and other attributes.

LSF requests resources from the resource connector by specifying the number of instances of a particular template that it requires to satisfy its demand. The resource connector uses the definitions in this file to map this demand into a set of allocation requests in Azure.

The default location for the file is

```
<LSF_TOP>/conf/resource_connector/azure/conf/azureprov_templates.json
```

**Description**

LSF requests resources from the resource connector by specifying the number of instances of a particular template that it requires to satisfy its demand. The resource connector uses the definitions in this file to map this demand into a set of allocation requests in Microsoft Azure.
Important: When you define templates, you must make sure that the attribute definitions that are presented to LSF exactly match the definitions that are provided by Microsoft Azure. If, for example, the attribute definition specifies hosts with ncpus=4, but the actual hosts that are returned by Microsoft Azure report ncpus=2, the demand calculation in LSF is not accurate.

Parameters

The file contains a JSON-defined list that is named templates. Each template in the list is an object that contains the following parameters:

Name
  The unique template name.

Attributes
  A list of attributes that represent the hosts in the template from the LSF point of view. LSF attempts to place its pending workload on hosts that match these attributes to calculate how many instances of each template to request.

Each attribute string in the list has the following format:
"attribute_name": ["attribute_type", "attribute_value"]

attribute_name
  An LSF resource name, for example, type or ncores.
  The attribute name must either be a built-in resource (such as type or ncpus),
  or defined in the Resource section in the lsf.shared file on the LSF master host.

attribute_type
  Can be either Boolean, String, or Numeric and must correspond to the
  corresponding resource definition in the lsf.shared file.

attribute_value
  The value of the resource that is provided by hosts. For Boolean resources,
  use 1 to define the presence of the resource and 0 to define its absence. For
  Numeric resources, specify a range that uses [min:max].

The following attributes have default values if they are not defined:

type
  The default value is given by the setting of the LSB_RC_DEFAULT_HOST_TYPE
  in the lsf.conf file. The default value of LSB_RC_DEFAULT_HOST_TYPE is
  X86_64.

ncpus
  Default value is 1.

ImageId
  The image name of the Azure Machine Image that LSF is installed on. The
  image name is used to launch virtual machine instances of this template.

MaxNumber
  The maximum number of available instances of this template. LSF never
  requests more than this number of instances for this template. Set this
  parameter to an appropriate value according to the instance quota of the
  project or the maximum capacity of the Microsoft Azure infrastructure.

resourceGroup
  The resource group name that the ImageId, virtualNetwork, and securityGroup
  belongs to.
storageAccountType

The storage account type is also referred to as the SkuName. It is required for account creation. It can have the following values: Standard_LRS and Premium_LRS. It maps to the VM disk type HDD (for Standard_LRS) or SSD (for Premium_LRS) when you create an instance on the Azure portal.

vmType

The instance size used to launch virtual instances. Note that, there are some restrictions between the storageAccountType and vmType attributes.

Tip: You can try to create one VM on the Azure portal, select VM disk type HDD or SSD, and list the available instance sizes that are supported by the storage account type.

virtualNetwork

The virtual network name which is used for instance interconnect. You must create the virtual network in the resource group configured in the resourceGroup attribute.

subnet

The subnet name (virtual private cloud) used to launch virtual instances. Use the subnet through which the instance can communicate with the LSF cluster. The subnet name must exist as a subnet of the virtual network configured in the virtualNetwork attribute.

sshPubKeyFile

The SSH public key file for the user account for virtual instances.

customScriptUri

URI that is injected into the instance after instance creation and executed during instance startup.

rootUserName

The login user account that is created on the instance with root permission. If not specified, the default login user is lsfuser.

securityGroup

The Azure security group that is applied to instances. This is optional if you have configured a default security group with the subnet attribute. If there is no default security group with the subnet and you don't specify the security group in template, an error is logged.

withPublicIP

It should be set to true if public IP is needed for instances. If not set or set to false, the public ip is not allocated for the instance.

instanceTags

A string representing a list of keys and their values. These N key value pairs are used to tag the instance (using Amazon instance tagging feature). If an instance is launched using TemplateA, it will be tagged with instanceTags defined in TemplateA.

If not specified, LSF still tags the newly launched instances with the following key-value pair: InstanceID = <<id of the instance created>>.
Example azureprov_templates.json file

```json
{
   "Templates": [
      {
         "templateId": "TemplateA",
         "Attributes": {
            "type": "["String", "X86_64"]",
            "ncpus": "["Numeric", "4"]",
            "mem": "["Numeric", "480"]",
            "maxmem": "["Numeric", "512"]",
            "azurehost": "["Boolean", "1"]",
            "zone": "["String", "southeastasia"]",
            "maxNumber": "1"
         },
         "resourceGroup": "lsf_rg",
         "imageId": "image-lsf-compute-rhel72",
         "storageAccountType": "PREMIUM_LRS",
         "vmType": "Standard_A0",
         "virtualNetwork": "lsf_vnet",
         "subnet": "lsf_compute_subnet1",
         "sshPubKeyFile": "/home/lsfadmin/lsf.pub",
         "securityGroup": "lsf_sg",
         "withPublicIP": false,
         "customScriptUri": "http://10.1.0.4/user_data.sh",
         "instanceTags": "group=LSF;project=Azure"
      }
   ]
}
```

The example defines a template that is named TemplateA. LSF attempts to place any pending workload on hypothetical hosts of type X86_64 with ncpus=4 and mem=480. If LSF successfully places some of its pending workload on \( N \) number of hosts, it requests \( N \) instances of TemplateA to the resource connector.

If demand is generated for this template, the connector attempts to start virtual machines that match the configured image and flavor in Microsoft Azure. If resource connector succeeds in running the launch command for any of the instances (even if there are fewer than requested), the connector informs LSF that it can use the allocated hosts.

The package list and log file is read from the environment variables `PACKAGES` and `LOGFILE`, the `UserScript` that is included in the instance is run during startup. The following example user script reads these variables and attempts to install the packages:

```bash
#!/bin/bash
yum install -y $(echo $PACKAGES| tr , " ") >> $LOGFILE 2>&1
```

softlayerprov_config.json

The `softlayerprov_config.json` file contains configuration parameters used by the resource connector to work with IBM Bluemix (formerly Soft Layer).

For example, you can configure the `softlayerprov_config.json` file to invoke remote IBM Bluemix services, such as creating virtual servers.

The default location for the file is

```bash
<LSF_TOP>/conf/resource.connector/softlayer/conf/softlayerprov_config.json
```
Parameters

The file is a JSON format tuple that contains the following parameter fields:

**LogLevel**

The log level for the host provider. Use the following log levels:
- INFO (the default level)
- DEBUG
- WARN
- ERROR
- FATAL

The log file path is `<LSF_TOP>/log/softlayer-provider.<host_name>.log`.

**SOFTLAYER_CREDENTIAL_FILE**

Absolute path to the file containing IBM Bluemix API credentials.

**SOFTLAYER_DOMAIN_NAME**

Sets the domain name of virtual hosts requested by the plugin.

**UserDataTimeout**

Sets the timeout value, in seconds, for IBM Bluemix API calls to set user metadata. The default is 120 seconds.

Example softlayerprov_config.json file

```json
{
  "LogLevel": "DEBUG",
  "SOFTLAYER_CREDENTIAL_FILE": "/opt/lsf/conf/resource_connector/softlayer/conf/credentials",
  "SOFTLAYER_DOMAIN_NAME": "hostfactory.com",
  "UserDataTimeout": 300
}
```

**softlayer_templates.json**

The softlayer_templates.json file defines the mapping between LSF resource demand requests and IBM Bluemix virtual servers.

The template represents a set of hosts that share some attributes such as the number of CPUs, the amount of available memory, the installed software stack, operating system, and other attributes.

LSF requests resources from the resource connector by specifying the number of instances of a particular template that it requires to satisfy its demand. The resource connector uses the definitions in this file to map this demand into a set of virtual server orders in IBM Bluemix.

The default location for the file is `<LSF_TOP>/conf/resource_connector/softlayer/conf/softlayer_templates.json`

**Description**

LSF requests resources from the resource connector by specifying the number of instances of a particular template that it requires to satisfy its demand. The resource connector uses the definitions in this file to map this demand into a set of allocation requests in IBM Bluemix.

**Important:** When you define templates, you must make sure that the attribute definitions that are presented to LSF exactly match the definitions that are
provided by IBM Bluemix. If, for example, the attribute definition specifies hosts with ncpus=4, but the actual hosts that are returned by IBM Bluemix report ncpus=2, the demand calculation in LSF is not accurate.

Parameters

The file contains a JSON-defined list that is named templates. Each template in the list is an object that contains the following parameters:

- **templateId**
  The unique template name.

- **attributes**
  A list of attributes that represent the hosts in the template from the LSF point of view. LSF attempts to place its pending workload on hosts that match these attributes to calculate how many instances of each template to request.

  Each attribute string in the list has the following format:
  "attribute_name": ["attribute_type", "attribute_value"]

  - **attribute_name**
    An LSF resource name, for example, type or ncores.
    The attribute name must either be a built-in resource (such as type), or defined in the Resource section in the lsf.shared file on the LSF master host.

  - **attribute_type**
    Can be either Boolean, String, or Numeric and must correspond to the corresponding resource definition in the lsf.shared file.

  - **attribute_value**
    The value of the resource that is provided by hosts. For Boolean resources, use 1 to define the presence of the resource and 0 to define its absence. For Numeric resources, specify a range that uses [min:max].

The following attributes have default values if they are not defined:

- **type**
  The default value is given by the setting of the `LSB_RC_DEFAULT_HOST_TYPE` in the lsf.conf file. The default value of `LSB_RC_DEFAULT_HOST_TYPE` is X86_64.

- **ncpus**
  Default value is 1.

- **imageId**
  The name of the IBM Bluemix image to be used for provisioning of virtual servers that LSF is installed on. The image name is used to launch virtual servers of this template.

- **postProvisionURL**
  An HTTPS URL that points to the post-provisioning script that you define.

- **userData**
  A string representing a list of keys and their values.

  The format of userData is
  
  `<key>=<value_list>;<key>=<value_list>`

  - **key**
    The key name of `userData` variable, such as packages, volume, zone, or templateName.
value_list
A list of userData values, separated by commas, for example, package1, package2.

After you define userData, it is divided into keys and values and exported as the virtual server environment variables. For example, if userData is defined as packages=M,N;logfile=X, it is exported as the following environment variables in the instances:
packages=M,N
logfile=X

These variables can be read by the provisioning script in the instance as keys: packages and logfile.

datacenter
Indicates the IBM Bluemix data center where virtual servers should be provisioned.

privateNetworkOnlyFlag
Must be set to false to enable public network on the virtual host, if the LSF cluster master is not an IBM Bluemix host that is accessible via the Bluemix internal VLAN.

vlanNumber
The name of the Bluemix VLAN to which the requested virtual server instances are to be connected. The VLAN must be defined in the data center specified by the datacenter parameter.

dedicatedAccountHostOnlyFlag
Specifies that a compute instance is to run only on hosts that have guests from the same account. If not specified, the current Bluemix default storage configuration is used.

maxNumber
An integer value determining the maximum number of virtual servers that can be provisioned using this template. This value limits the number of dynamic hosts that can be requested by LSF.

localDiskFlag
If true, the storage attached to the ordered virtual server instances are local disks. If false, the storage is SAN disks. If not specified, the current Bluemix default storage configuration will be used.

useHourlyPricing
If set to true, virtual server instances are ordered with hourly billing. If set to false, virtual server instance are ordered with monthly billing. If not specified, the current Bluemix default billing mode is used.

Example softlayerprov_templates.json file
{
  "templates": [ 
  {
    "templateId": "Template-1",
    "maxNumber": 10,
    "attributes": {
      "type": ["String", "X86_64"],
      "ncpus": ["Numeric", "1"],
      "ncores": ["Numeric", "1"],
      "nram": ["Numeric", "4096"],
      "softlayerhost": ["Boolean", "1"],
      "customattr": ["String", "somedata"]
    },
  },
}
"imageId": "lsfslavehosts",
"datacenter": "tor01",
"vlanNumber": "882",
"useHourlyPricing": true,
"localDiskFlag": false,
"privateNetworkOnlyFlag": false,
"dedicatedAccountHostOnlyFlag": false,
"postProvisionURL": "https://169.55.139.117/post-prov.sh",
"userData": "one=2;foo=bar;zone=vm-1"
},
{
"templateId": "Template-2",
"maxNumber": 10,
"attributes": {
  "type": ["String", "X86_64"],
  "ncpus": ["Numeric", "4"],
  "ncores": ["Numeric", "1"],
  "nram": ["Numeric", "8192"]
},
"imageId": "lsfslavehosts",
"datacenter": "tor01",
"vlanNumber": "882",
"useHourlyPricing": true,
"localDiskFlag": false,
"privateNetworkOnlyFlag": false,
"dedicatedAccountHostOnlyFlag": false,
"postProvisionURL": "https://169.55.139.117/post-prov.sh",
"userData": "one=2;foo=bar;zone=vm-1"
}
]
}

This example defines a template named Template-2. LSF attempts to place any pending workload on hosts of type X86_64 with ncpus=4 and nram=8192. If LSF successfully places some of its pending workload on N number of hosts, it requests N instances of Template-2 to the resource connector. If demand is generated for this template, the resource connector attempts to launch virtual machines, up to the defined maximum of 10, matching the configured imageId on Bluemix. If it succeeds in running the launch command for any of the instances (even if there are fewer than requested), the connector informs LSF that it can use the allocated hosts.

Note: Any of the parameters defined in the attributes section in the template can be used in the select string to select a specific template.

For example, bsub -R "select[ncpus ==1]" creates a host with the first template.
Notices

This information was developed for products and services offered in the U.S.A.

IBM® may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user’s responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

© Copyright IBM Corp. 2015, 2017
IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Intellectual Property Law
Mail Station P300
2455 South Road,
Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM’s future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or
imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

### Trademarks

IBM, the IBM logo, and ibm.com® are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at [http://www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

### Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

#### Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

#### Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.
Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Privacy policy considerations

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.
