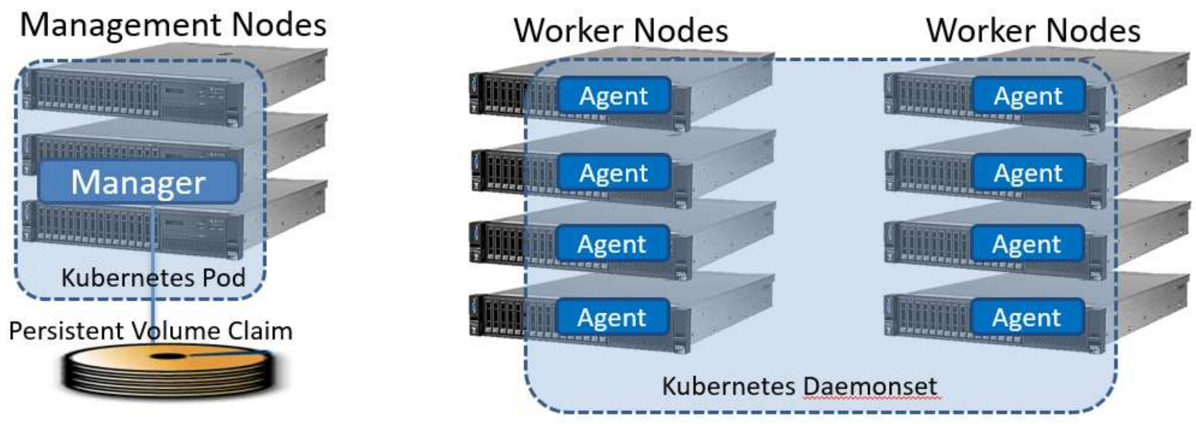
IBM Spectrum Computing Cloud Pak Quickstart Guide

This documents how to install the IBM Spectrum Computing Cloud Pak on IBM Cloud Private 3.1.1 and above. This type of installation adds new features to Kubernetes for running jobs, including the ability to perform the following:

- Schedule complex jobs
- Run parallel jobs
- Prioritize jobs
- Schedule GPU jobs with consideration of CPU/GPU topology
- Share resources equitably among many users

This installation takes an existing IBM Cloud Private cluster and deploys the components needed to provide enhanced scheduling. Existing IBM Spectrum LSF users that want to have better service capabilities can try the alternative installation that adds Kubernetes to a portion of an existing LSF cluster.

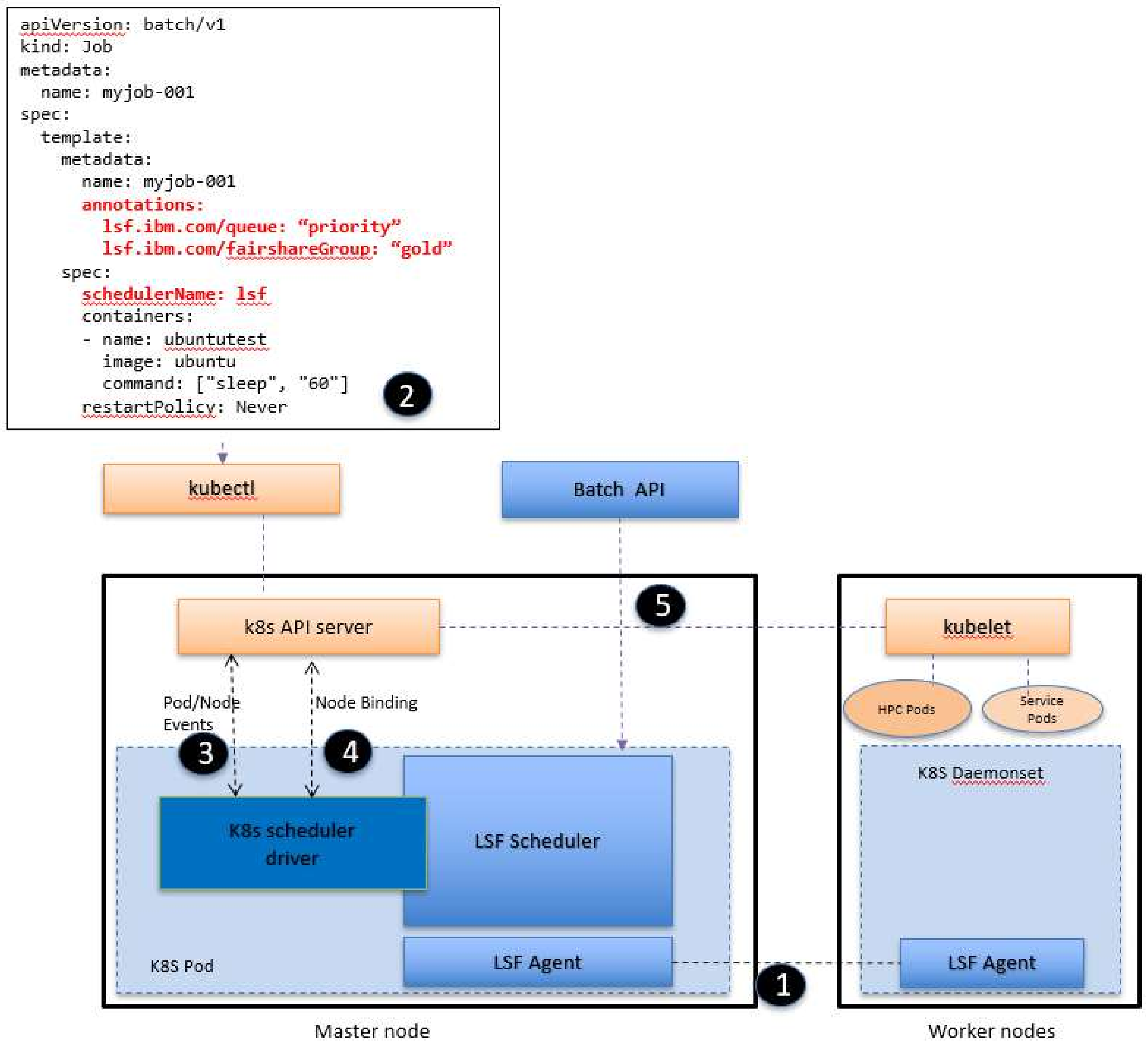
Once the Cloud Pak is deployed you will have a cluster like the following example:



Agents will be deployed on all the worker nodes. These are deployed as a daemonset and gather information for the Manager to use for job execution and data collection. One of the management nodes will run the Manager pod. The Manager pod runs the scheduling and resource management processes needed for the jobs. The Manager pod uses the persistent volume claim to store job data.

Overview

We have taken the core LSF scheduling technology and integrated into Kubernetes to combine the expressive power of the Kubernetes API with the rich resource sharing and load-balancing technology that is at the heart of LSF. Here is how it works:



```
apiVersion: batch/v1
kind: Job
metadata:
  name: myjob-001
spec:
  template:
    metadata:
      name: myjob-001
      annotations:
        lsf.ibm.com/queue: "priority"
        lsf.ibm.com/fairshareGroup: "gold"
    spec:
      schedulerName: lsf
      containers:
      - name: ubuntutest
        image: ubuntu
        command: ["sleep", "60"]
      restartPolicy: Never
```

1. The LSF Scheduler components are packaged into containers and a Helm chart is provided to deploy into the IBM Cloud Private environment.
2. Users submits workload into K8S API via kubectl. To get the LSF Scheduler to be aware of the pod the "schedulerName" field must be set, otherwise the pod will be scheduled by the default scheduler. Scheduler directives can be specified using annotations in the pod.
3. In order to be aware of the status of pods and nodes, the LSF Scheduler uses a driver that listens to Kubernetes API server and translates pod requests into jobs in the LSF Scheduler.
4. Once the LSF Scheduler makes a policy decision on where to schedule the pod, the driver will bind the pod to specific node.

5. The Kubelet will execute and manages pod lifecycle on target nodes in the normal fashion.

The LSF Scheduler also supports jobs submitted from the native batch CLI, however this is available in a different package.

Installation and Evaluation Steps

Perform the following steps to evaluate the IBM Spectrum Computing Cloud Pak:

1. Install the prerequisites
2. Deploy the Cloud Pak
3. Verify the Cloud Pak deployment
4. Deploy the jobs
5. Going Further

The following sections will explain how to complete the steps.

Install the Prerequisites

The following are needed to evaluate the Cloud Pak:

- An installation of IBM Cloud Private 3.1.1 and up with:
  - At least two machines with one dedicated “worker” node.
  - The “cloudctl” command line installed.
  - Optionally the “kubectl” and “helm” command line interfaces.
- Dedicated persistent volume for the deployment to use.

At least two machines are required for the evaluation. During the installation of IBM Cloud Private the `cluster/hosts` file controls the role of the machines in the cluster. A sample `cluster/hosts` file might look like the following:

```
[master]
10.10.10.10

[worker]
10.10.10.20
10.10.10.21
10.10.10.22
```

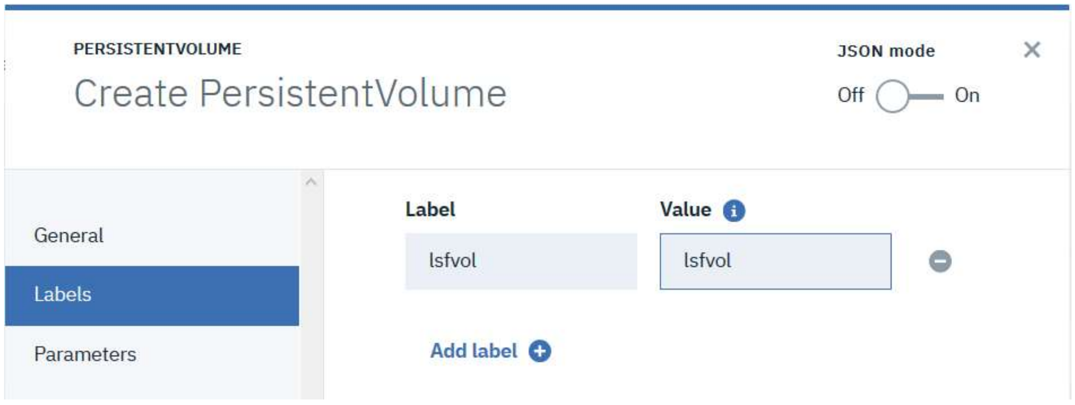
The `[master]` section defines which machines will run the IBM Cloud Private management functions. One of these machines will run the Manager pod. The `[worker]` section defines the list of machines that will run the services and jobs.

Once the IBM Cloud Private cluster is installed, the command line interfaces can be installed. For more instructions on how to install these commands, refer to the following:

- Installation of the “cloudctl” cli:
  https://www.ibm.com/support/knowledgecenter/en/SSBS6K_3.1.2/manage_cluster/icp_cli.html
- Installation of the “kubectl” cli:
  https://www.ibm.com/support/knowledgecenter/en/SSBS6K_3.1.2/manage_cluster/install_kubectl.html
- Installation of the “helm” cli:
  https://www.ibm.com/support/knowledgecenter/en/SSBS6K_3.1.2/app_center/create_helm_cli.html

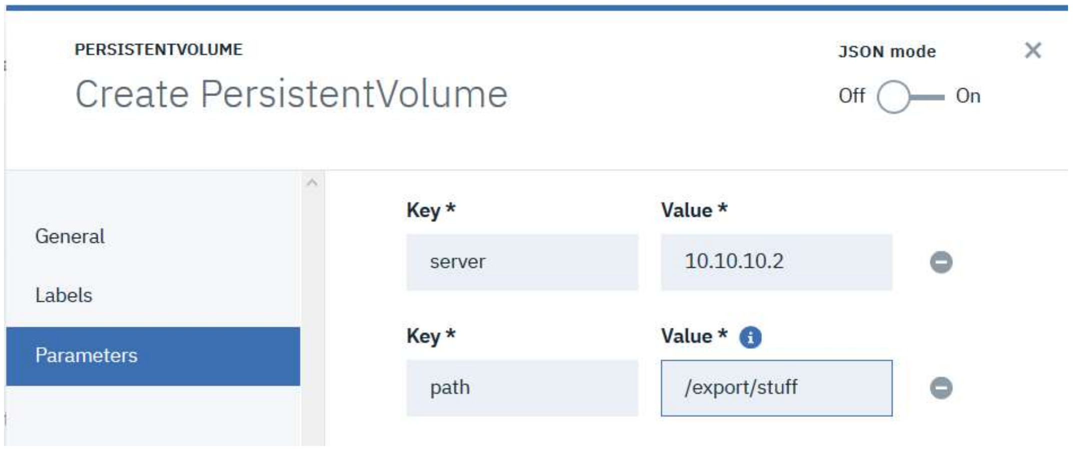
Note: The links are for 3.1.2. Instructions may vary depending on version.

A Persistent Volume (PV) is needed to store the configuration and workload data. This should be in a well-known location as there is a need to back up the data and change the configuration. Create a PV within the GUI by navigating to *Platform* -> *Storage* and clicking on *Create PersistentVolume*. Use ReadWriteOnce, or ReadWriteMany for the access mode. Set the Reclaim Policy to Retain or Recycle depending on how you want the data to be handled. Set the name and size. Set the Labels as shown follows:



Label	Value
lsfvol	lsfvol

Set the Parameters for your storage. The parameters for an NFS server are shown below. The “server” is the IP address of the NFS server. The “path” should be set to the directory the NFS server is exporting, for example:



Key *	Value *
server	10.10.10.2
path	/export/stuff

Once done click Create. The PersistentVolume list should have the volume, and it should be free.

Deploy the Cloud Pak

Deployment of the Cloud Pak has three steps:

1. Load the Cloud Pak into IBM Cloud Privates local catalog.
2. (Optional) Exclude some hosts from the scheduler
3. Deploy the Helm chart.

Step 1: Load the Cloud Pak into IBM Cloud Privates local catalog

The “cloudctl” cli is used to import IBM Spectrum Computing Cloud Pak into IBM Cloud Private. Log in to begin the process with:

```
$ cloudctl login -a {URL of the ICP master e.g. https://10.10.10.70:8443} --skip-ssl-
validation -u admin
```

It will prompt for a password. This is the same password that was set during installation and used to login to the GUI. It will then prompt for the namespace. Testing has been done with the Default namespace. A sample Pod Security Policy and ClusterRole is provided with the chart.

Next login to the local repository with:

```
$ docker login mycluster.icp:8500
```

If IBM Cloud Private was installed with a different cluster name, then use that instead of “`mycluster.icp`”. Once logged in, import the catalog by running the following command and the installation file you downloaded from the IBM site:

```
$ cloudctl catalog load-archive --archive ibm-spectrum-computing-
1.0.0.tgz
```

After the command finishes it may take a few minutes before the synchronization finishes.

Step 2: (Optional) Exclude Some Hosts from the Scheduler

You may wish to not allow some machines to be used for HPC, or AI workloads. HPC and AI workloads tend to be very CPU intensive and can negatively affect other processes running on the physical machine. Label the machines you want to reserve for performance sensitive services, so that they will not be used for HPC or AI workloads. For example:

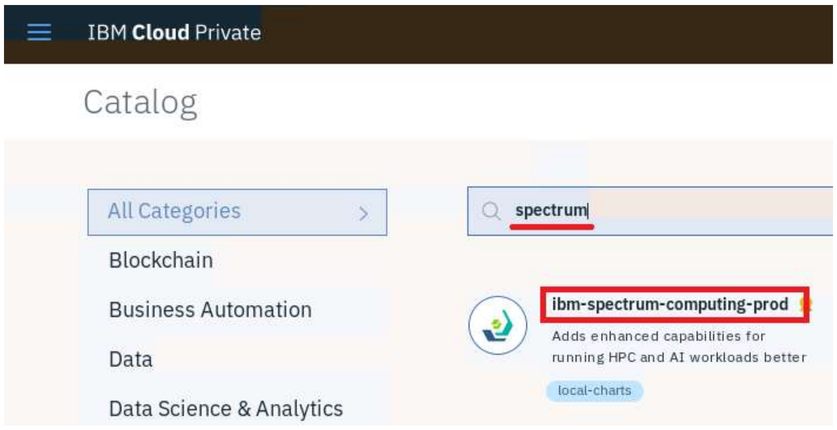
```
$ kubectl get nodes --show-labels

$ kubectl label nodes {Node name from above} excludelsf=ok
```

The value of the label is not important and can be anything you like. You may use another label name but have to provide that name in the next step.

Step 3: Deploy the Helm chart

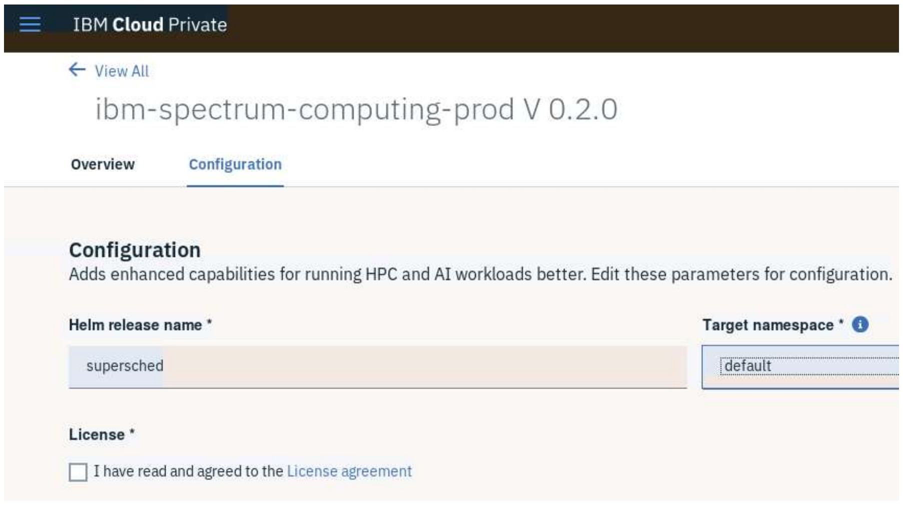
Log in to the IBM Cloud Private GUI to deploy the Cloud Pak. Click on the "Catalog" and in the search bar type "spectrum". You should see the following:



Select "ibm-spectrum-computing-prod" to continue the deployment. Documentation and sample commands and configuration will be displayed. Click "Configure" to customize the deployment. Do the following:

1. Set the name of the Helm release.
2. Choose the Target namespace. Select "default" for this technical preview.
3. Read through and accept the licenses.

You will see something like the following:



Configuration

Adds enhanced capabilities for running HPC and AI workloads better. Edit these parameters for configuration.

Helm release name * : supersched

Target namespace * : default

License *

☐ I have read and agreed to the License agreement

NOTE: The version shown is the chart version and not the IBM Spectrum Computing version, and higher versions of IBM Cloud Private will look different.

Check the Storage Configuration in the Parameter Section.

If you wanted to exclude some hosts. Provide the label used to designate machines that should not be part of the cluster.

NOTE: When using an NFS persistent volume make sure to set the GID of the persistent volume.

If the NFS server is exporting "/export/stuff", check the group for /export/stuff for example,

```
# ls -la /export/stuff
total 8
drwxr-xr-x 2 mblack mygroup 4096 Mar 25 14:55 .
```

Here the group is "mygroup". Make sure the filesystem group read, write and execute bit is set by running:

```
# chmod 775 /export/stuff
```

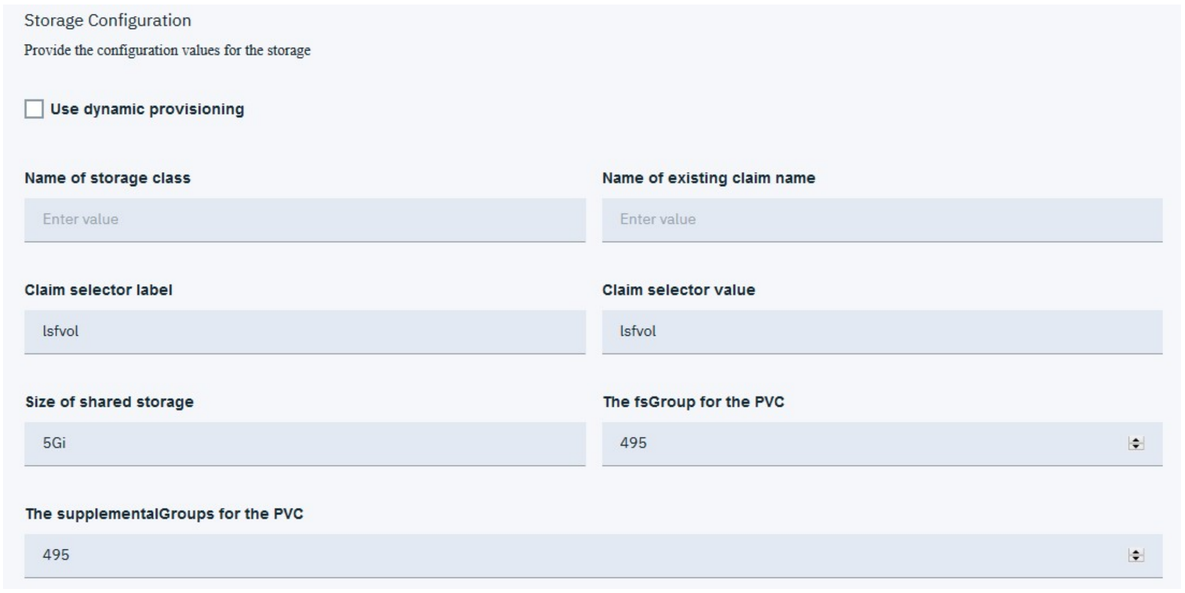
Determine the GID of the filesystem group, in this case "mygroup", by running:

```
# getent group |grep mygroup
mygroup:x:495:
```

Here, the GID of the "mygroup" group is 495. Use that value for the:

- The fsGroup for the PVC
- The supplementalGroups for the PVC

For example:



Storage Configuration

Provide the configuration values for the storage

☐ Use dynamic provisioning

Name of storage class: Enter value	Name of existing claim name: Enter value
Claim selector label: lsfvol	Claim selector value: lsfvol
Size of shared storage: 5Gi	The fsGroup for the PVC: 495
The supplementalGroups for the PVC: 495	

NOTE: You NFS server will have a different path and group. Use your values.

For worker nodes that have GPUs and are not using the Nvidia docker runtime it is necessary to define the *nvidiapath*. This is the path that the kubelet is using to access the Nvidia driver. It may look something like: */var/lib/kubelet/device-plugins/nvidia-driver/init*

Click on "Install". The pods will be deployed on the worker machines and one of the manager machines.

View the helm release to see what is being deployed.

Verify the Cloud Pak Deployment

Use the following procedures to verify that the Cloud Pak is functioning correctly. If you have not done so already login with the "cloudctl" cli, for example:

```
$ cloudctl login -a {URL of the ICP master e.g. https://10.10.10.10:8443} --skip-ssl-
validation -u admin
```

List the helm deployments with the following command:

```
$ helm list --tls |grep spectrum-computing
supersched                    1              Mon Apr 25 15:07:56 2019
DEPLOYED        ibm-spectrum-computing-prod-1.1.0  default
```

This shows that the chart is deployed and that it is called: "supersched". Now check the daemonset by running the following command:

```
$ kubectl get daemonsets --namespace {Namespace used to deploy chart}
NAME                  DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE
SELECTOR                         AGE
supersched-..-agent   3         3         2       3            2           node-
role.kubernetes.io/worker=true   23m
```

The desired number of pods is 4, but only 3 are running. List the pods to see which one is not working:

```
# kubectl get pods
NAME                               READY   STATUS              RESTARTS   AGE
supersched-…-agent-7mbjj           1/1     Running             0          3d20h
supersched-…-agent-h4ht2           1/1     Running             0          3d20h
supersched-…-agent-njqll           0/1     ContainerCreating   0          3d20h
supersched-…-master-56b55d8-84gcj  1/1     Running             0          3d20h
```

Check the logs of the container that is not in Running state with the following command:

```
$ kubectl logs supersched-ibm-spectrum-computing-prod-agent-njqll
```

Or get more information about the pod with the following command:

```
$ kubectl describe pod supersched-ibm-spectrum-computing-prod-agent-njqll
```

Connect to the `ibm-scheduler` pod for the following tests with the following command:

```
$ kubectl exec -ti supersched-ibm-spectrum-computing-prod-master-56b55d8-84gcj
bash
```

```
LSF POD [root@master /]#
```

NOTE: The prompt changed to indicate you are operating in a pod.

Try some LSF commands to see the cluster state for example:

```
LSF POD [root@master /]# lsid
IBM Spectrum LSF Standard Edition 10.1.0.0, Jun 24 2019
Copyright International Business Machines Corp. 1992, 2016.
US Government Users Restricted Rights - Use, duplication or disclosure restricted
by GSA ADP Schedule Contract with IBM Corp.

My cluster name is myCluster
My master name is lsfmaster
```

This tells you that the manager processes are running. Next check the workers state by running the following command:

```
LSF POD [root@lsfmaster /]# lsload -w
HOST_NAME               status  r15s   r1m  r15m   ut    pg  ls    it   tmp   swp   mem
worker-10-10-10-20          ok   0.5   0.7   1.1   1%   0.0   0  3e+5  373G  3.4G 251.6G
worker-10-10-10-21          ok   0.7   0.4   1.6   1%   0.0   0    39  267G  3.7G 251.5G
worker-10-10-10-22          ok   0.7   0.1   0.9   1%   0.0   0    39  166G  3.4G 251.5G
lsfmaster                   ok   1.3   3.0   5.3   8%  2e+3   0  3e+5 4128G  3.3G 125.6G
```

and

```
LSF POD [root@lsfmaster /]# bhosts -w
HOST_NAME          STATUS          JL/U    MAX  NJOBS    RUN  SSUSP  USUSP    RSV
lsfmaster          closed_Full      -       0     0       0      0      0      0
worker-10-10-10-20 ok               -       -    30      30      0      0      0
worker-10-10-10-21 ok               -       -    37      37      0      0      0
worker-10-10-10-22 ok               -       -    39      39      0      0      0
```

NOTE: The host names that the scheduler uses are not the pod names, they are a representation of the IP address of the host that is running the pod, and not the pod IP.

The lsfmaster is closed_Full deliberately to prevent pods being scheduled on the management nodes.

The machines STATUS should be ok. If the worker machine state is ok, they are ready to accept jobs. If the cluster just started or you reconfigured the cluster, it may take a while for all machines to be ok.

Deploy Jobs

Once installed the Cloud Pak enables new options for jobs run through Kubernetes. These options are needed for running High Performance Computing (HPC) applications, and large AI jobs. They extend the pod specification with annotations to define which scheduling and placement policies to use. To enable the enhanced features the pod specification must have the schedulerName set to “lsf” for example:

```
spec.template.spec.schedulerName:  lsf
```

The new features for kubernetes jobs that this provides includes the following:

- Job Priority
- Application Profiles
- Fair sharing of resources

- GPU Management
- Parallel Jobs (More details below)

The following table shows the new annotations along with the LSF equivalent.

Pod Spec Field	**Description**	**LSF Job Submission Option**
`*.metadata.name`	A name to assign to the job	Job Name (-J)
`++.lsf.ibm.com/dependency`	A job dependency expresion	Job Dependency (-w)
`++.lsf.ibm.com/project`	A project name to assign to job	Project Name (-P)
`++.lsf.ibm.com/application`	An application profile to use	Application Profile (-app)
`++.lsf.ibm.com/gpu`	The GPU requirements for the job	GPU requirement (-gpu)
`++.lsf.ibm.com/queue`	The name of the job queue to run the job in	Queue (-q)
`++.lsf.ibm.com/jobGroup`	A job group to assign to job	Job Group (-g)
`++.lsf.ibm.com/fairshareGroup`	The fairshare group to assign the job to	Fairshare Group (-G)
`++.lsf.ibm.com/user`	The user to run the application as, and for accounting	Job submission user
`++.lsf.ibm.com/serviceClass`	The service class to apply to the job	Service class (-sla)
`++.lsf.ibm.com/reservation`	The resources to reserve prior to running the job	Advanced Reservation (-U)
`*.spec.containers[].resources.requests.memory`	The amount of memory to reserve for the job	Memory Reservation (-R "rusage[mem=...]")
`*.spec.schedulerName`	Set to "lsf"	N/A

NOTE:

*** - The pod specification files should be prefaced with *spec.template*.**

++ - The pod specification files should be prefaced with *spec.template.metadata.annotations*.

For information on the annotations and their meanings refer to the following:

https://www.ibm.com/support/knowledgecenter/SSWRJV_10.1.0/lsf_welcome/lsf_kc_cluster_ops.html

These capabilities are accessed by modifying the pod specifications for jobs. Below is a minimal example:

```
apiVersion: batch/v1
kind: Job
metadata:
  name: myjob-001
spec:
  template:
    metadata:
      name: myjob-001
    spec:
      schedulerName: lsf        # This directs scheduling to the LSF Scheduler
      containers:
      - name: ubuntutest
        image: ubuntu
        command: ["sleep", "60"]
        resources:
          requests:
            memory: 5Gi
      restartPolicy: Never
```

This example enables Kubernetes to use **lsf** as the job scheduler. The LSF job scheduler can then apply its policies to choose when and where the job will run.

Additional parameters can be added to the pod yaml file to control the job. The example below shows how to use the additional annotations:

```
apiVersion: batch/v1
kind: Job
metadata:
  name: myjob-001
spec:
  template:
    metadata:
      name: myjob-001
      # The following annotations provide additional scheduling
      # information to better place the pods on the worker nodes
      # NOTE:  Some annotations require additional LSF configuration
      annotations:
        lsf.ibm.com/project: "big-project-1000"
        lsf.ibm.com/queue: "normal"
        lsf.ibm.com/jobGroup: "/my-group"
        lsf.ibm.com/fairshareGroup: "gold"
    spec:
      # This directs scheduling to the LSF Scheduler
      schedulerName: lsf
      containers:
      - name: ubuntutest
        image: ubuntu
        command: ["sleep", "60"]
      restartPolicy: Never
```

In the previous example, the annotations provide the LSF scheduler more information about the job and how it should be run.

Users that submit a job through Kubernetes typically are trusted to run services and workloads as other users. For example, the pod specifications allow the pod to run as other users:

```
apiVersion: batch/v1
kind: Job
metadata:
  name: myjob-uid1003-0002
spec:
  template:
    metadata:
      name: myjob-uid1003-0002
    spec:
      schedulerName: lsf
      containers:
      - name: ubuntutest
        image: ubuntu
        command: ["id"]
      restartPolicy: Never
      securityContext:
        runAsUser: 1003
        fsGroup: 100
        runAsGroup: 1001
```

In the previous example, the pod would run as UID 1003 and produce the following output:

```
uid=1003(billy) gid=0(root) groups=0(root),1001(users)
```

Note the GID and groups. Care should be taken to limit who can create pods. Alternatively, LSF applications can be used to allow the administrator to predefine pod specification file.

Parallel Jobs

The chart includes a new Custom Resource Definition (CRD) for parallel jobs. This simplifies the creation of parallel jobs in kubernetes. The ParallelJob CRD describes the resource requirements for parallel jobs with multiple tasks on K8s. The ParallelJob controller daemon is responsible to create seperate Pods for each task described in the ParallelJob CRD.

The CRD supports both job-level and task-level scheduling terms which can satisfy common scheduling needs over all the Pods in the same job or individual need for each Pod. At the same time, one can also specify all the Pod Spec policies for the Pod defined in the ParallelJob CRD.

Job Level Terms

ParallelJob CRD supports the following job-level terms to describe the resource requirements apply for all the Pods in the same parallel job.

- **spec.description**: the human readable description words attached to the parallel job
- **spec.resizable**: the valid values are "true" or "false", which determines whether the Pods in the parallel job should be co-scheduling together. Specifically, a resizable job can be started with a

few Pods got enough resources, while a non-resizable job must get enough resources for all of the Pods before starting any Pods.

- **spec.headerTask**: typical parallel jobs (e.g. Spark, MPI, Distributed Tensorflow) run a "driver" task to co-ordinate or work as a central sync point for the left tasks. This term can be used to specify the name of such "driver" task in a parallel job. It will make sure the header task can be scheduled and started before or at the same time with other non-header tasks.
- **spec.placement**: this term supports multiple sub-terms which can satisfy various task distribution policies, such as co-allocating multiple tasks on the same host or zone, or evenly distribute the same number of tasks across allocated hosts. This term can be defined in both job-level and task-group level.

- **spec.priority**: this term is used to specify job priority number which can rank the parallel job with other jobs submitted by the same user. The default maximum number can be supported by LSF is 100.

Currently, this term supports the following placement policies. The example defines a "same" policy in job-evel to enforce all the tasks belong to the parallel job co-allocated to the nodes in the same zone.

```
sameTerm: node | rack | zone
spanTerms:
- topologyKey: node
  taskTile: #tasks_per_topology
```

To use the topology keys, you must define the following host based resources in your LSF configuration files. Add the following highlighted lines to the lsf.shared file. See the "Change the Scheduler Configuration" section below for more information.

```
Begin Resource
RESOURCENAME  TYPE    INTERVAL INCREASING  DESCRIPTION
...
kube_name     String  ()       ()          (Kubernetes node name)
rack_name     String  ()       ()          (Kubernetes node rack name)
zone_name     String  ()       ()          (Kubernetes node zone name)
End Resource
```

Modify the lsf.cluster host entries and add the rack_name and zone_name e.g.

```
Begin   Host
HOSTNAME   model  type  server  RESOURCES
...
ICPHost01  !      !     1       (kube_name=172.29.14.7 rack_name=blade1
zone_name=Florida)
End Host
```

Task Level Terms

The tasks are grouped by the common resource requirements of replicas.

- **spec.taskGroups[].spec.replica**: this term defines the number of tasks in current task group
- **spec.taskGroups[].spec.placement**: this term shares the same syntax with the one defined at job level. The second task group in the example defines an alternative "span" like placement policy, which can either put 4 replicas across two nodes or on the same node.

- **spec.taskGroups[].spec.template.spec**: the Pod Spec shares the same syntax supported by your K8s cluster. For example, you can specify the nodeSelector to filter node labels during scheduling.

LSF Specific Annotations

The annotations defined at job-level can support job control extensions with prefix of "lsf.ibm.com/" listed in here. The resource requirements conflict of the following extensions are described as follows.

- **lsf.ibm.com/gpu**: Number of GPUs to be requested on each host (-gpu). This term will be ignored when the Pod explicitly request nvidia.com/gpu resource in ParallelJob CRD.

Submit ParallelJob CRD

The following example submission script describes a parallel job which have two replicas (tasks) in total.

```
apiVersion: ibm.com/v1alpha1
kind: ParallelJob
metadata:
  name: double-tasks-parallel
  namespace: default
  labels:
    lable1: example2
spec:
  name: double-tasks-parallel
  description: This is a parallel job with two tasks to be running on the same
node.
  headerTask: group0
  priority: 100
  schedulerName: lsf
  taskGroups:
  - metadata:
      name: group0
    spec:
      placement:
        sameTerm: node
        spanTerms:
        - topologyKey: node
          taskTile: 2
      replica: 2
      template:
        spec:
          containers:
          - args:
            image: ubuntu
            command: ["sleep", "30"]
            name: task1
            resources:
              limits:
                cpu: 1
```

Sample jobs may also be found on GitHub: https://github.com/IBMSpectrumComputing/lsf-kubernetes

Monitor ParallelJob CRD

Use the following command to monitor the status of a parallel job submitted using ParallelJob CRD. It will give the Job Status together with the counters of its Pods in various Pod phases as Task Status.

When the Job is in Pending status, the command shows the Job Pending Reason of corrosponding LSF control job.

```
$ kubectl describe pj
Name:          parallel-job
Namespace:     default
Annotations:   <none>
API Version:   ibm.com/v1alpha1
```

```
Kind:          ParallelJob
...
...
Status:
  Job Pending Reason:  "New job is waiting for scheduling;"
  Job Status:          Pending
  Task Status:
    Unknown:    0
    Failed:     0
    Pending:    5
    Running:    0
    Succeeded:  0
```

The LSF control job ID is attached as a Pod label named lsf.ibm.com/jobId on each Pod. Several special Pod labels are attached to record the information of its parallel job belongs to.

```
$ kubectl describe po
Name:                double-tasks-parallel-kflb9
Namespace:           default
Priority:            0
PriorityClassName:   <none>
Node:                <none>
Labels:              controller-uid=de751862-9114-11e9-864a-3440b5c56250
                     lsf.ibm.com/jobId=2762
                     parallelJob.name=double-tasks-parallel
                     parallelJob.taskGroup.index=1
                     parallelJob.taskGroup.name=group1
Annotations:         lsf.ibm.com/pendingReason: "New job is waiting for
scheduling;"
...
...
```

Going Further

Additional documentation and examples are available from:

https://github.com/IBMSpectrumComputing/lsf-kubernetes

Questions can also be emailed to: LSF-Inquiry@ca.ibm.com

Or posted on our slack channel. To join go here:

https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/W1559b1be149d_43b0_881e_9783f38faaff/page/Connect

Change the Scheduler Configuration

The scheduler stores policy configuration in the persistent volume claim used by the manager pod.

Additional configuration for the queues and fairshareGroups is stored in configMaps. The default configuration can be changed, and information about the file formats is available from:

https://www.ibm.com/support/knowledgecenter/SSWRJV_10.1.0/lsf_welcome/lsf_kc_cluster_ops.html

What follows is an overview of how to change both the configuration stored in the persistent volume claim, and the configMaps.

Changing Configuration Files

Changing the scheduler configuration requires:

- Connecting to the manager pod
- Changing the configuration file(s)
- Reconfiguring the scheduler

To connect the manager pod use the following procedure:

1. Locate the master pod by looking for **ibm-spectrum-computing-prod-master** in the list of pods e.g.

   ```
   $ kubectl get pods |grep ibm-spectrum-computing-prod-master

   lsf-ibm-spectrum-computing-prod-master-56b55d6dc8-84gcj    1/1       Running    0
   3d19h
   ```

2. Connect to the management pod e.g.

   ```
   $ kubectl exec -ti lsf-ibm-spectrum-computing-prod-master-56b55d6dc8-84gcj bash
   ```

The configuration files are in: */opt/ibm/lsfsuite/lsf/conf*

The directory has the following files in it:

```
conf/cshrc.lsf
conf/profile.lsf
conf/hosts
conf/lsf.conf
conf/lsf.cluster.myCluster
conf/lsf.shared
conf/lsf.task
conf/lsbatch/myCluster/configdir/lsb.users       ← Exposed as ConfigMap
conf/lsbatch/myCluster/configdir/lsb.nqsmaps
conf/lsbatch/myCluster/configdir/lsb.reasons
conf/lsbatch/myCluster/configdir/lsb.hosts
conf/lsbatch/myCluster/configdir/lsb.serviceclasses
conf/lsbatch/myCluster/configdir/lsb.resources
conf/lsbatch/myCluster/configdir/lsb.modules
conf/lsbatch/myCluster/configdir/lsb.threshold
conf/lsbatch/myCluster/configdir/lsb.applications
conf/lsbatch/myCluster/configdir/lsb.globalpolicies
```

```
conf/lsbatch/myCluster/configdir/lsb.params
conf/lsbatch/myCluster/configdir/lsb.queues
```
🡐 Exposed as ConfigMap

NOTE: Do not directly edit the configmap files, otherwise you will loose your changes.

Find the file you want to change and modify it.

After changing the configuration files(s) it is necessary to trigger the scheduler to re-read the configuration. This will not affect running or pending workload. From within the management pod do the following:

1. Run the command to reconfigure the base

   ```
   LSF POD [root@lsfmaster /]# lsadmin reconfig

   Checking configuration files ...

   No errors found.

   Restart only the master candidate hosts? [y/n] y
   Restart LIM on <lsfmaster> ...... done
   ```

   To reconfigure the base on all nodes use:

   ```
   LSF POD [root@lsfmaster /]# lsadmin reconfig all
   ```

2. Run the command to re-read the schduler configuration:

   ```
   LSF POD [root@lsfmaster /]# badmin mbdrestart

   Checking configuration files ...

   There are warning errors.

   Do you want to see detailed messages? [y/n] y
   Apr    22    13:14:49    2019    22437    4    10.1    orderQueueGroups():    File
   /opt/ibm/lsfsuite/lsf/conf/lsbatch/myCluster/configdir/lsb.queues:      Priority
   value <20> of queue <night> falls in the range of priorities defined for the
   queues that use the same cross-queue fairshare/absolute priority scheduling
   policy. The priority value of queue <night> has been set to 1
   ---------------------------------------------------------
   No fatal errors found.
   Warning: Some configuration parameters may be incorrect.
            They are either ignored or replaced by default values.

   Do you want to restart MBD? [y/n]
   ```

   Here we see there is an error. The initial configuration will not have errors, but it is instructive to see what they might look like.

3. If errors are seen, correct them, and retry the command to check that the errors have been corrected.

Changing the ConfigMap Files

Two configuration files are exposed as configMaps. They are:

- **lsb.users** - This contains the users and user groups for configuring fairshare
- **lsb.queues** - This contains the queue definitions

They can be edited in the GUI, or using the following commands:

```
$ kubectl get configmap
```

This will list all the config maps. Look for ones containing the string `ibm-spectrum-computing-prod`.

```
$ kubectl edit configmap lsf-ibm-spectrum-computing-prod-queues
```

NOTE: You will see additional metadata associated with the configmap. Do not change this.

Changes to the configMaps will be automatically applied to the cluster. Errors in the configMaps will cause the scheduler to revert to a default configuration. To check for errors use the procedure in the above section to test for errors, but remember that changes to the **lsb.users** and **lsb.queues** have to be done by editing the configmap.

Log Files

The log files can provide useful information for troubleshooting problems. The log files to look at are in the Manager pod, however by default they are forwarded to STDOUT so that running

```
$kubectl logs  {Name of Pod}
```

will display the logs. Sometimes this is not convenient.

. To access the logs directly, use the following procedure:

1. Get the name of the Manager pod:

   ```
   $ kubectl get pods --namespace {Namespace used to deploy chart}
   ```
   Look for the pod names containing "`ibm-spectrum-computing-prod-master`",

2. Connect to the manager pod:

   ```
   $ kubectl exec -ti {Pod name from above} bash
   ```

3. Turn off log file forwarding:

   ```
   LSF POD [root@master /]# touch /tmp/debug
   ```

4. Go to the log directory and look at logs:

   ```
   LSF POD [root@master /]# cd /opt/ibm/lsfsuite/lsf/log
   ```

```
LSF POD [root@master /]# more kubebridge.lsfmaster.log
```

An example of a configuration error might look like the following:

```
LSF POD [root@master log]# more kubebridge.lsfmaster.log
Log file created at: 2019/03/27 14:38:40
Running on machine: lsfmaster
Binary: Built with gc go1.11.2 for linux/amd64
Log line format: [IWEF]mmdd hh:mm:ss.uuuuuu threadid file:line] msg
E0327 14:38:40.125268     376 jobhandler.go:308] Unable to submit the job
to lsf. cmd<bsub -q normal -g my-group -P big-project-1000 -G bestshare
-J  default/myjob-001-6hljx  -ext  kube[default/myjob-001-6hljx]  sleep
1000000> error<Bad or empty job group name. Job not submitted.>
```

To correct this problem, and ones like it, modify the job pod specification and make sure LSF is configured with the right fair share groups, job groups, users etc.

Adding Users and Groups

Some of the workload policies need to be aware of the user running the job. For these policies to function, it is necessary to provide the usernames, UIDs, and GIDs typically found in `/etc/passwd` and `/etc/group` files. The **`add-users-groups.sh`** sample script is provided to import the information from the host machine's operating system. To import the users and groups, go a Linux machine that has both users and groups configured (such as LDAP/NIS), and has access to the persistent volume, then run the following commands:

```
$ cd {Location of PV mount}
$ cd lsf/conf
$ add-users-groups.sh
```

This script will call **getent** to gather all the users and group information and generate two files:

- passwd.append
- group.append

The master container, upon detecting these files will import the passwd and group information. Once this is done the fairshare policies for users can be configured.

Reconfiguring the Cluster

When the configuration files are changed it is necessary to reconfigure the cluster. This can be done within the cluster by running the appropriate commands, or alternatively by using the **`trigger-reconfig.sh`** helper script. Inside the manager container the script is located here:

```
/opt/ibm/lsfsuite/lsf/conf/trigger-reconfig.sh
```

Within the persistent volume it is located here:

{PV mount point}`/ lsf / conf / trigger-reconfig.sh`

Backups

Configuration and state information is stored in the persistent volume claim. Backups of that data should be performed periodically. The state information can become stale very fast as users work is submitted and finished. Some job state data will be lost for jobs submitted between the last backup and current time.

NOTE: A reliable filesystem is critical to minimize job state loss.

Dynamic provisioning of the persistent volume is discouraged because of the difficulty in locating the correct resource to backup. Pre-creating a persistent volume claim, or labeling a persistent volume, for the deployment to use provides the easiest way to locates the storage to backup.

Restoring from a backup will require restarting the manager processes. Use the procedure below to reconfigure the entire cluster after restoring files.

1. Locate the master pod by looking for ***ibm-spectrum-computing-prod-master*** in the list of pods e.g.
   ```
   $ kubectl get pods |grep ibm-spectrum-computing-prod-master

   lsf-ibm-spectrum-computing-prod-master-56b55d6dc8-84gcj    1/1       Running    0
   3d19h
   ```
2. Connect to the management pod e.g.
   ```
   $ kubectl exec -ti lsf-ibm-spectrum-accel-prod-master-56b55d6dc8-84gcj bash
   ```
3. Run the command to re-read the configuration files
   ```
   LSF POD [root@lsfmaster /]# lsadmin reconfig
   LSF POD [root@lsfmaster /]# badmin mbdrestart
   ```
4. Wait for a minute and try some commands to see if the cluster is functioning okay e.g.
   ```
   LSF POD [root@lsfmaster /]# lsid
   IBM Spectrum LSF Standard Edition 10.1.0.0, Jun 24 2019
   Copyright International Business Machines Corp. 1992, 2016.
   US Government Users Restricted Rights - Use, duplication or disclosure restricted
   by GSA ADP Schedule Contract with IBM Corp.

   My cluster name is myCluster
   My master name is lsfmaster
   ```
   Command should report the software versions and manager hostname.
   ```
   LSF POD [root@lsfmaster /]# bhosts
   # bhosts
   HOST_NAME        STATUS     JL/U    MAX  NJOBS    RUN  SSUSP  USUSP    RSV
   ```

```
w10-10-10-10      unreach      -     -      0      0      0      0      0
w10-10-10-11      closed       -     -      0      0      0      0      0
w10-10-10-12      ok           -     -      0      0      0      0      0
lsfmaster         ok           -     -      0      0      0      0      0
```

Host status should be **ok**.

```
LSF POD [root@lsfmaster /]# bqueues
QUEUE_NAME     PRIO STATUS       MAX  JL/U JL/P JL/H NJOBS  PEND   RUN  SUSP
priority        43  Open:Active   -    -    -    -     0     0     0     0
normal          30  Open:Active   -    -    -    -     0     0     0     0
idle            20  Open:Active   -    -    -    -     0     0     0     0
night            1  Open:Inact    -    -    -    -     0     0     0     0
```

Queues should be open.

Copyright and trademark information