CDP Private Cloud

Migrating Fair Scheduler to Capacity Scheduler for CDP Private Cloud Base

Date published: 2019-08-22
Date modified: 2022-04-08

https://docs.cloudera.com/

Legal Notice

Contents

Fair Scheduler to Capacity Scheduler migration overview

You must migrate from Fair Scheduler to Capacity Scheduler when migrating your cluster to Cloudera. The migration process involves automatically converting certain Fair Scheduler configuration to Capacity Scheduler configuration and manual fine tuning after the migration.

In Cloudera, Capacity Scheduler is the default and supported scheduler. You have to migrate from Fair Scheduler to Capacity Scheduler when migrating from CDH to Cloudera Base on premises. It does not matter what CDH version you are migrating from, or what Cloudera version you are migrating to, the scheduler migration steps remain the same.

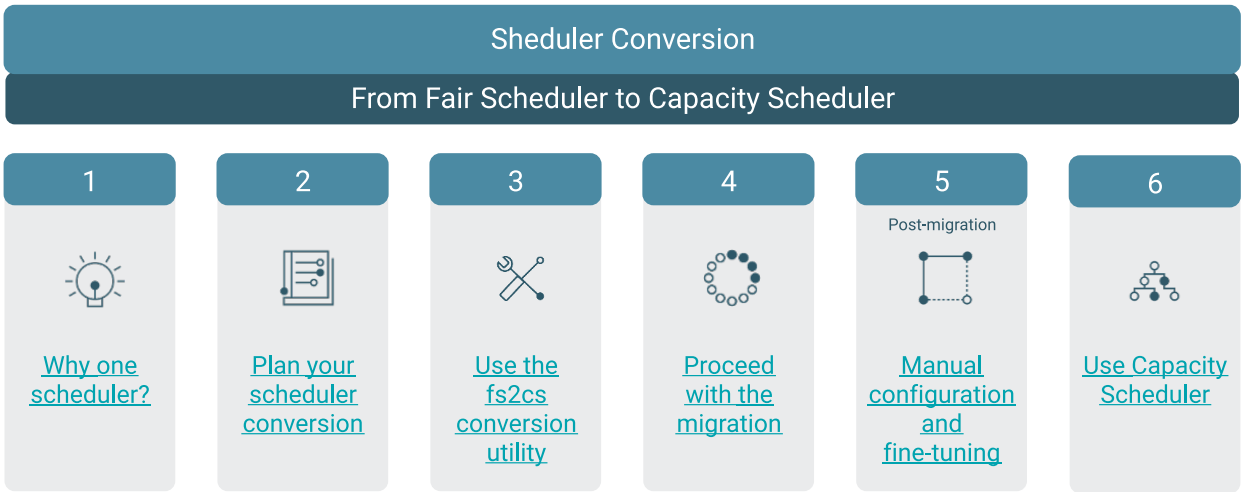
The fs2cs conversion utility is used to convert the Fair Scheduler configuration into Capacity Scheduler. From Cloudera Base on premises 7.1.6, the fs2cs conversion utility converts from weight resource allocation mode into weight resource allocation mode by default. In lower versions the utility converts to relative resource allocation mode by default.

Important: The features of Capacity Scheduler are not the same as the features of Fair Scheduler. Hence, the fs2cs conversion utility cannot convert every Fair Scheduler configuration into a Capacity Scheduler configuration. After the automatic conversion and once the migration is completed, you must manually tune the scheduler configurations to ensure that the resulting scheduling configuration fits your organization's internal goals and SLAs.

The scheduler migration process includes migrating the YARN settings from Fair Scheduler to Capacity Scheduler. When converting your Fair Scheduler to Capacity Scheduler, you will perform the following steps:

1. Learn about why only one scheduler, the Capacity Scheduler, is supported in Cloudera.
2. Plan your scheduler conversion: Read about the conversion limitations, and ensure you understand what properties are converted by the conversion utility and what properties require manual conversion and fine tuning once the migration is complete.
3. Convert Fair Scheduler to Capacity Scheduler: Download the Fair Scheduler configuration, and use the fs2cs conversion utility to convert the Fair Scheduler configuration into Capacity Scheduler configuration. After conversion, upload the new configuration files.
4. Migrate all of your data to Cloudera.
5. Manually configure and fine-tune the Capacity Scheduler so it fits your organization's goals.
6. Start using the Capacity Scheduler.

For more information, click on the step that interests you:



Related Information

Comparison of Fair Scheduler with Capacity Scheduler

Managing and allocating cluster resources using Capacity Scheduler

Planning your scheduler migration

Before starting the scheduler migration, you must learn about what Fair Scheduler configuration can be converted into a Capacity Scheduler configuration prior to the migration, and what configuration requires manual configuration and fine-tuning.

The features of Capacity Scheduler are not exactly the same as the features of Fair Scheduler. Hence, the fs2cs conversion utility cannot convert every Fair Scheduler configuration into a Capacity Scheduler configuration. You must learn about what properties are auto-converted and what requires manual configuration. In addition, there are Fair Scheduler features that do not have an equivalent feature in Capacity Scheduler.

Scheduler migration limitations

There are some hard limitations on converting a Fair Scheduler configuration into a Capacity Scheduler configuration as these two schedulers are not equivalent. Learning about these major limitations can help you understand the difficulties you might encounter after the scheduler migration.

The features and configurations of Capacity Scheduler differ from the features and configurations of Fair Scheduler resulting in scheduler migration limitations. These limitations sometimes can be overcome either by manual configuration, fine-tuning or some testing, but in some cases there is no workaround.

Note: This is not a complete list. It only contains the scheduler migration limitations that most commonly cause issues.

Static and dynamic leaf queues cannot be created on the same level

If you have a parent queue defined in capacity-scheduler.xml file with at least a single leaf queue, it is not possible to dynamically create a new leaf under this particular parent.

Resolved: Weight mode and Dynamic Auto Child Creation is supported from Cloudera Runtime 7.1.6. In weight mode there can be static and dynamic queues on the same level.

Placement rules and mapping rules are different

Placement rules (used in Fair Scheduler) and mapping rules (used in Capacity Scheduler) are very different, therefore auto-conversion is not possible. You manually have to configure placement rules and mapping rules once the upgrade from CDH to Cloudera is completed. There are multiple reasons for this. The following are the most substantial differences:

- In Fair Scheduler you can use special placement rules like "default" or "specified" which are completely absent in Capacity Scheduler.
- In Fair Scheduler you can set a "create" flag for every rule. Mapping rules do not support this.
- In Fair Scheduler in case of nested rules the "create" flag is interpreted for both rules. This is not true in Capacity Scheduler.
- If a rule can return a valid queue in Fair Scheduler, it proceeds to the next rule. Capacity Scheduler, on the other hand, returns “root.default”.

Resolved: In Cloudera Runtime 7.1.6 a new JSON-based placement rule format and a new placement rules evaluation engine were introduced. They resolve many of the previous placement rule migration limitations.

Mixed resource allocation mode is not supported

In Capacity Scheduler all queues can be either absolute, relative or weighted. That means that hybrid mix of absolute and weighted queues is not supported in Capacity Scheduler.

The capacity value of dynamic queues is fixed

In Fair Scheduler, fair shares are recalculated each time a new queue is created. In contrast, Capacity Scheduler assigns a predefined percentage value for dynamically created queues.

This predefined percentage can be changed, but it is fixed until the scheduler is reconfigured. Once this value reaches 100, the next dynamic queue will be created with the value 0. For example, if the value is set to 25.00, then the fifth queue under the same parent will have a capacity of 0.

The following is an example of how you can convert the Fair Scheduler queue weights to Capacity Scheduler queue capacity (percentage relative to its parents) :

Table 1: Weight conversion example

Queue Path	Weight	Capacity Scheduler equivalent (capacity) yarn.scheduler.capacity.<*QUEUE-PATH*>.capacity
root	1	100%
root.default	10	25%
root.users	30	75%
root.users.alice	1	33.333%
root.users.bob	1	33.333%
root.users.charlie	1	33.334%

In Cloudera Runtime 7.1.5 and lower versions the fs2cs conversion utility ensures that all percentages of direct children under one parent queue add up exactly to 100.000%, as it is demonstrated in the table. For example, all queues under root.users: root.users.alice + root.users.bob + root.users.charlie = 100.000%.

Weights are converted into percentage-based capacities the following way: On queue-level root, there are 2 queues: default and users. Because it is specified as 10 + 30 weights (40 altogether), 1 “unit of weight” is 2.5%. This is why root.default has 25% and root.users has 75% of the capacity. This calculation can be applied to all queue-levels.

Resolved: Weight mode and Dynamic Auto Child Creation is supported from Cloudera Runtime 7.1.6. and the fs2cs conversion utility converts into weight mode by default.

Placement rules migration

Placement rules migration is part of the Fair Scheduler to Capacity Scheduler migration process. Learn about the limitations of this migration and how you can overcome them.

Migrating to Cloudera Base on premises 7.1.5 or lower

Placement rules (used in Fair Scheduler) and mapping rules (used in Capacity Scheduler) are very different, therefore auto conversion of placement rules into mapping rules are not possible. You manually have to configure placement rules and mapping rules on the upgrade from CDH to Cloudera is completed.

The following are the most substantial limitation:

- In Fair Scheduler you can use special placement rules like "default" or "specified" which are completely absent in Capacity Scheduler.
- In Fair Scheduler you can set a "create" flag for every rule. Mapping rules do not support this.
- In Fair Scheduler in case of nested rules the "create" flag is interpreted for both rules. This is not true in Capacity Scheduler.
- If a rule can return a valid queue in Fair Scheduler, it proceeds to the next rule. Capacity Scheduler, on the other hand, returns “root.default”.

For more information see *Fair Scheduler features and conversion details*.

Migrating to Cloudera Base on premises 7.1.6 or higher

In Cloudera Runtime 7.1.6 and later releases there is a new placement engine that supports a new JSON-based mapping rule format. These new mapping rules eliminated many previous mapping rule limitations caused by the transitioning from Fair Scheduler to Capacity Scheduler. Note that in weight mode more limitations are resolved than in percentage mode.

The new placement engine can be thought of as a superset of Fair Scheduler and Capacity Scheduler placement evaluation logic. This means two things:

- Most items described in Fair Scheduler's <queuePlacementPolicy> section can be converted into Capacity Scheduler with some minor exceptions.
- Full backward compatbility with the old queue-mapping rule format.

The following are the most substantial differences between the old placement and the new mapping rules:

- Mapping rules can be created and managed using YARN Queue Manager.
- The rules are described in JSON, however, this is transparent to the user in CDP. The generated JSON can be viewed in Cloudera Manager as part of the Capacity Scheduler Configuration Advanced Configuration Snippet (Safety Valve) setting.
- You can configure what should happen when a rule cannot place the application to the target queue. There are three options: proceed to the next rule, reject the submission, place the application in the default queue. Previously Fair Scheduler proceeded to the next rule in such cases as it was more complicated to achieve a 'reject' or 'place to default queue' behavior.
- New policies (mapping actions) are available: specified, defaultQueue, and reject.
- The create flag was introduced: Non-existing queues are only created dynamically if it is enabled.

The following limitations remains when transitioning from the Fair Scheduler placement rules to the Capacity Scheduler mapping rules:

- When using nested mapping rules, it is not possible to define two separate create flags for the parent and the leaf queue.
- Fair Scheduler performs a strict validation whether a rule in the chain is reachable or not. The placement engine in Capacity Scheduler does not perform such a validation.

For more information, see *Auto-converted Fair scheduler properties*, *Fair Scheduler features and conversion details*, and *Configuring Placement Rules*.

Auto-converted Fair Scheduler properties

The fs2cs conversion utility automatically converts certain Fair Scheduler properties into Capacity Scheduler properties. Reviewing the list of auto-converted properties enables you to verify the conversion and plan the manual fine-tuning that requires to be done after the migration is completed.

Migrating to Cloudera Base on premises 7.1.6 or higher

Table 2: Queue resource-quota related features

Property	Description
Pre-created hierarchical queues.	The same queue hierarchy is achieved after conversion.
<weight>	Weight: The steady fair share of a queue. The queue.capacity property will be set with the same ratio.
<maxAMShare>	Maximum AM share: Limits the fraction of the queue’s fair share that can be used to run application masters
<maxRunningApps>	Maximum running apps: Limits the number of apps from the queue to run at once

Property	Description
<maxContainerAllocation>	Maximum container allocation: Maximum amount of resources a queue can allocate for a single container.
<schedulingPolicy>	Scheduling policy of a queue (for example, how submitted applications are ordered over time). It is converted with some limitations. For more information, see *Fair Scheduler features and the conversion details*.
<aclSubmitApps> <aclAdministerApps>	ACL settings: List of users and/or groups that can submit apps to the queue or can administer a queue.
maximum-applications	Specifies the maximum number of concurrent active applications at any one time in the queue.
maximum-am-resource-percent	Specifies the maximum percentage of resources in the cluster which can be used to run application masters for the queue.
acl_submit_applications	Specifies the ACL which controls who can submit applications to the given queue.
acl_administer_queue	Specifies the ACL which controls who can administer applications in the given queue.
ordering-policy	Specifies the queue ordering policies to FIFO or fair on the given queue.

Table 3: Global scheduling settings

Property	Description
yarn.scheduler.fair.allow-undeclared-pools	Allow undeclared pools. Sets whether new queues can be created at application submission time.
yarn.scheduler.fair.sizebasedweight	Size based weight. Whether to assign shares to individual apps based on their size, rather than providing an equal share to all apps regardless of size.
<queueMaxAppsDefault>	Queue max apps default: Sets the default running app limit for all queues.
<queueMaxAMShareDefault>	Default max AM share: Sets the default AM resource limit for queue.
yarn.scheduler.fair.locality.threshold.node	Locality threshold node: For applications that request containers on particular nodes, the number of scheduling opportunities since the last container assignment to wait before accepting a placement on another node.
yarn.scheduler.fair.locality.threshold.rack	Locality threshold rack: For applications that request containers on particular racks, the number of scheduling opportunities since the last container assignment to wait before accepting a placement on another rack.
yarn.scheduler.fair.max.assign	Maximum assignments: If assignmultiple is true and dynamic.max.assign is false, the maximum amount of containers that can be assigned in one heartbeat.
yarn.scheduler.fair.assignmultiple	Assign multiple: Whether to allow multiple container assignments in one heartbeat.
yarn.resourcemanager.scheduler.monitor.enable	Allows higher-priority applications to preempt lower-priority applications.

Property	Description
yarn.scheduler.capacity.maximum-am-resource-percent	Specifies the maximum percentage of resources in the cluster which can be used to run application masters.
<userMaxAppsDefault>	Default maximum running applications.
<user name="..."> <maxRunningApps>...</maxRunningApps></user>	Maximum running applications per user.
yarn.scheduler.fair.user-as-default-queue	Whether to use the username associated with the allocation as the default queue name. Weight mode: This behavior is simulated with a mapping rule (in fact, even in Fair Scheduler, this is translated into a placement rule internally): `{ "type":"user", "matches":"*", “parentQueue”: “root”, "policy":"user", “create”: true, "fallbackResult":"skip" }` For information about percentage mode, see *Fair Scheduler features ad conversion details*.

Table 4: Preemption

Property	Description
yarn.scheduler.fair.preemption	Fair Scheduler preemption turned on. After the conversion capacity Scheduler preemption is turned on by default using the default values.
<allowPreemptionFrom>	Per-queue preemption disabled. After the conversion the same queue preemption disabled by default.
yarn.scheduler.fair.waitTimeBeforeKill	Wait time before killing a container
disable_preemption	Disables preemption of application containers submitted to a given queue.

Table 5: Placement rules

Fair Scheduler placement rules	Description	Conversion details
create="false" or "true"	Disable or enable creating a queue dynamically in YARN. This option cannot be specified on the following placement rule policies: • reject • setDefaultQueue • defaultQueue	Weight mode: This flag is fully supported, except for nested rules, where you can define a single “create” flag only. Therefore, “true/false” and “false/true” cannot be set. Relative mode: Partially supported. A managed parent queue must be chosen as a parent. The flag has no effect on regular parent queues. For more information about managed parent queue, see Managed Parent Queues.
<rule name="specified"/>	If a user has submitted the application by specifying a queue name (other than the “default” queue), then this rule will be successful. Hence the remaining set of rules won't be executed.	Supported in both weight and percentage mode.

Fair Scheduler placement rules	Description	Conversion details
<rule name="primaryGroup"/>	If the submitted user's(userA) primary group name (groupA) exists, submit to groupA.	The matching policy is called primaryGroup.
<rule name="secondaryGroupExistingQueue" />	If the submitted user's(userA) secondary group name (groupB) exists, submit to groupB.	The matching policy is called secondaryGroup.
<rule name="nestedUserQueue">	Depending on the nested rule, this places the job to the following queues: • root.[primaryGroup].[userName] • root.[secondaryGroup].[userName] • root.[queuePath].[userName]	Supported by Capacity Scheduler. The three possible policies are (depending on the outer rule): • primaryGroupUser • secondaryGroupUser • user with a parentQueue set explicitly.
<rule name="default" queue="qName"/>	Places the application into the default queue called "root.default" or to a user-specific one denoted by the "queue" attribute.	The default rule has a matching policy called defaultQueue. If "root.default" is not the intended default queue, then two approaches are possible: • Use the setDefaultQueue policy to change "root.default", then apply defaultQueue. • Use the custom policy with the policy string being set to the target queue.

Migrating to Cloudera Base on premises 7.1.5 or lower

Table 6: Queue resource-quota related features

Property	Description
Pre-created hierarchical queues.	The same queue hierarchy is achieved after conversion.
<weight>	Weight: The steady fair share of a queue. The queue.capacity property will be set with the same ratio.
<maxAMShare>	Maximum AM share: Limits the fraction of the queue's fair share that can be used to run application masters
<maxRunningApps>	Maximum running apps: Limits the number of apps from the queue to run at once
<maxContainerAllocation>	Maximum container allocation: Maximum amount of resources a queue can allocate for a single container.
<schedulingPolicy>	Scheduling policy of a queue (for example, how submitted applications are ordered over time). It is converted with some limitations. For more information, see *Fair Scheduler features and the conversion details*.
<aclSubmitApps> <aclAdministerApps>	ACL settings: List of users and/or groups that can submit apps to the queue or can administer a queue.
maximum-applications	Specifies the maximum number of concurrent active applications at any one time in the queue.
maximum-am-resource-percent	Specifies the maximum percentage of resources in the cluster which can be used to run application masters for the queue.
acl_submit_applications	Specifies the ACL which controls who can submit applications to the given queue.

Property	Description
acl_administer_queue	Specifies the ACL which controls who can administer applications in the given queue.
ordering-policy	Specifies the queue ordering policies to FIFO or fair on the given queue.

Table 7: Global scheduling settings

Property	Description
yarn.scheduler.fair.allow-undeclared-pools	Allow undeclared pools. Sets whether new queues can be created at application submission time.
yarn.scheduler.fair.sizebasedweight	Size based weight. Whether to assign shares to individual apps based on their size, rather than providing an equal share to all apps regardless of size.
<queueMaxAppsDefault>	Queue max apps default: Sets the default running app limit for all queues.
<queueMaxAMShareDefault>	Default max AM share: Sets the default AM resource limit for queue.
yarn.scheduler.fair.locality.threshold.node	Locality threshold node: For applications that request containers on particular nodes, the number of scheduling opportunities since the last container assignment to wait before accepting a placement on another node.
yarn.scheduler.fair.locality.threshold.rack	Locality threshold rack: For applications that request containers on particular racks, the number of scheduling opportunities since the last container assignment to wait before accepting a placement on another rack.
yarn.scheduler.fair.max.assign	Maximum assignments: If assignmultiple is true and dynamic.max.assign is false, the maximum amount of containers that can be assigned in one heartbeat.
yarn.scheduler.fair.assignmultiple	Assign multiple: Whether to allow multiple container assignments in one heartbeat.
yarn.resourcemanager.scheduler.monitor.enable	Allows higher-priority applications to preempt lower-priority applications.
yarn.scheduler.capacity.maximum-am-resource-percent	Specifies the maximum percentage of resources in the cluster which can be used to run application masters.

Table 8: Preemption

Property	Description
yarn.scheduler.fair.preemption	Fair Scheduler preemption turned on. After the conversion capacity Scheduler preemption is turned on by default using the default values.
<allowPreemptionFrom>	Per-queue preemption disabled. After the conversion the same queue preemption disabled by default.
yarn.scheduler.fair.waitTimeBeforeKill	Wait time before killing a container
disable_preemption	Disables preemption of application containers submitted to a given queue.

Fair Scheduler features and conversion details

Certain Fair Scheduler properties cannot be auto-converted by the fs2cs conversion utility. Review the list of these properties and if they are supported in Capacity Scheduler and by Queue Manager UI to learn how you can configure them.

Migrating to Cloudera Base on premises 7.1.6 or higher

Table 9: Queue resource-quota related features

Property	Description	Conversion information
<minResources>	Minimum resources the queue is entitled to.	Partially supported in Capacity Scheduler. Ignored by the fs2cs conversion utility. Not supported by Queue Manager UI.
<maxResources>	Maximum amount of resources that will be allocated to a queue.	There is an equivalent feature in Capacity Scheduler. Ignored by the fs2cs conversion utility. For each queue, max-capacity will be set to 100%. Supported by Queue Manager UI. For more information, see Set user limits within a queue.
<maxChildResources>	Maximum amount of resources that can be allocated to an ad hoc child queue.	There is an equivalent feature in Capacity Scheduler. Ignored by the fs2cs conversion utility. Its value can be two distinct percentages (vcore/memory) or an absolute resources, but the leaf-queue-template only accepts a single percentage. Supported by Queue Manager UI.

Table 10: Global scheduling settings

Property	Description	Conversion information
yarn.scheduler.fair.max.assign	Dynamic maximum assign	There is an equivalent feature in Capacity Scheduler. Fine-tuning of the following three properties are required: • yarn.scheduler.capacity.per-node-heartbeat.multiple-assignments-enable • yarn.scheduler.capacity.per-node-heartbeat.maximum-container-assignments • yarn.scheduler.capacity.per-node-heartbeat.maximum-offswitch-assignments Not supported by Queue Manager UI.

Property	Description	Conversion information
yarn.scheduler.fair.user-as-default-queue	User as default queue	Relative mode:A placement rule needs to be created. 1. Create a queue, such as "root.users" and enable Dynamic Auto Child Creation for it (make it a Managed Parent Queue). 2. Create the following placement rule: `{ "type":"user", "matches":"*", “parentQueue”: “root.users”, "policy":"user", “create”: true, "fallbackResult":"skip" }` The following limitations apply: • It is not possible to have "root" as a parent for dynamically created queues. • "root.users" queue cannot have static leafs. Those are queues that always exist and are created manually. For information about weight mode see *Auto-converted Fair Scheduler properties*. Supported by Queue Manager UI.

Table 11: Preemption

Property	Description	Conversion information
yarn.scheduler.fair.preemption.cluster-utilization-threshold	The utilization threshold after which preemption kicks in.	There is an equivalent feature in Capacity Scheduler: yarn.resourcemanager.monitor.capacity.preemption.max_ignored_over_capacity. It specifies the resource usage threshold over its configured capacity that a queue must meet before it is eligible for preemption. Supported by Queue Manager UI.
minSharePreemptionTimeout	The number of seconds the queue is under its minimum share before it will try to preempt containers to take resources from other queue.s	Not supported in Capacity Scheduler.
fairSharePreemptionTimeout	The number of seconds the queue is under its fair share threshold before it will try to preempt containers to take resources from other queues.	Partially supported in Capacity Scheduler. This can be achieved by using the following configurations together: • yarn.resourcemanager.monitor.capacity.preemption.natural_termination_factor • yarn.resourcemanager.monitor.capacity.preemption.max_wait_before_kill Supported by Queue Manager UI.

Property	Description	Conversion information
fairSharePreemptionThreshold	The fair share preemption threshold for the queue.	Partially supported in Capacity Scheduler. This can be achieved by using the following configurations together: • yarn.resourcemanager.monitor.capacity.preemption.natural_termination_factor • yarn.resourcemanager.monitor.capacity.preemption.max_wait_before_kill Supported by Queue Manager UI.

Table 12: Placement rules

Fair Scheduler placement rules	Description	Conversion information
create="false" or "true"	Disable or enable creating a queue dynamically in YARN. This option can be specified on all rules.	Partially supported in Capacity Scheduler. Use the Capacity Scheduler Dynamic Queue Mappings policies: • u:%user:[managedParentQueueName].[queueName] • u:%user:[managedParentQueueName].%user • u:%user:[managedParentQueueName].%primary_group • u:%user:[managedParentQueueName].%secondary_group Supported by Queue Manager UI.
<rule name="specified"/>	If a user has submitted the application by specifying a queue name (other than the “default” queue), then this rule will be successful. Hence the remaining set of rules won't be executed.	Not supported in Capacity Scheduler.
<rule name="primaryGroupExistingQueue"/>	If submitted user’s(userA) primary group name (groupA) exists, submit to groupA.	There is an equivalent placement rule in Capacity Scheduler: <value>u:%user:%primary_group</value> Supported by Queue Manager UI.
<rule name="secondaryGroupExistingQueue" />	If submitted user’s(userA) secondary group name (groupA) exists, submit to groupA.	There is an equivalent placement rule in Capacity Scheduler: <value>u:%user:%secondary_group</value> Supported by Queue Manager UI.
<rule name="nestedUserQueue">	Depending on the nested rule, this places the job to the following queues: • root.[primaryGroup].[userName] • root.[secondaryGroup].[userName] • root.[queuePath].[userName]	Not supported in Capacity Scheduler.
<rule name="default" queue=”qName”/>	Fall back policy by which rule will fall back to queue named in the property ‘queue’ or the “default“ queue if no queue property is specified (if all matches fail).	There is an equivalent placement rule in Capacity Scheduler: <value>u:%user:default</value> Supported by Queue Manager UI.

Migrating to Cloudera Base on premises 7.1.5 or lower

Table 13: Queue resource-quota related features

Property	Description	Conversion information
<minResources>	Minimum resources the queue is entitled to.	Partially supported in Capacity Scheduler. Ignored by the fs2cs conversion utility. Not supported by Queue Manager UI.
<maxResources>	Maximum amount of resources that will be allocated to a queue.	There is an equivalent feature in Capacity Scheduler. Ignored by the fs2cs conversion utility. For each queue, max-capacity will be set to 100%. Supported by Queue Manager UI.
<maxChildResources>	Maximum amount of resources that can be allocated to an ad hoc child queue.	There is an equivalent feature in Capacity Scheduler. Ignored by the fs2cs conversion utility. Its value can be two distinct percentages (vcore/memory) or an absolute resources, but the leaf-queue-template only accepts a single percentage. Supported by Queue Manager UI.
<schedulingPolicy>	Scheduling policy of a queue (for example, how submitted applications should be ordered over time). .	There is an equivalent feature in Capacity Scheduler. Manual fine tuning might be necessary. **Note:** if DRF is used anywhere in Fair Scheduler, then the converted configuration utilizes DRF everywhere and it is not possible to place a queue with “Fair” policy under one which has “DRF” enabled. Supported by Queue Manager UI.

Table 14: Global scheduling settings

Property	Description	Conversion information
<user name="..."> <maxRunningApps>...</maxRunningApps></user>	Maximum running apps per user	There is an equivalent feature in Capacity Scheduler. Fine-tuning of the following three properties are required: • Maximum apps per queue • User limit percent • User limit factor Supported by Queue Manager UI. For more information about user limits, see Set user limits within a queue.
<userMaxAppsDefault>	Default maximum running apps	Not supported in Capacity Scheduler.

Property	Description	Conversion information
yarn.scheduler.fair.max.assign	Dynamic maximum assign	There is an equivalent feature in Capacity Scheduler. Fine-tuning of the following three properties are required: • yarn.scheduler.capacity.per-node-heartbeat.multiple-assignments-enable • yarn.scheduler.capacity.per-node-heartbeat.maximum-container-assignments • yarn.scheduler.capacity.per-node-heartbeat.maximum-offswitch-assignments Supported by Queue Manager UI.
yarn.scheduler.fair.user-as-default-queue	User as default queue	There is a very similar feature in Capacity Scheduler. Perform the following steps: 1. Create a queue, such as root.users and enable the suto-create-child-queue setting for it. 2. Use the following placement rule: "u%user:%user" The following restrictions apply: • It is not possible to have root as a parent for dynamically created queues. • root.users cannot have static leafs, that is, queues that are defined in the capacity-scheduler.xml file. For more information, see the *Placement Rules* table. Supported by Queue Manager UI.

Table 15: Preemption

Property	Description	Conversion information
yarn.scheduler.fair.preemption.cluster-utilization-threshold	The utilization threshold after which preemption kicks in.	There is an equivalent feature in Capacity Scheduler: yarn.resourcemanager.monitor.capacity.preemption.max_ignored_over_capacity. It specifies the resource usage threshold over its configured capacity that a queue must meet before it is eligible for preemption. Supported by Queue Manager UI.
minSharePreemptionTimeout	The number of seconds the queue is under its minimum share before it will try to preempt containers to take resources from other queue.s	Not supported in Capacity Scheduler.
fairSharePreemptionTimeout	The number of seconds the queue is under its fair share threshold before it will try to preempt containers to take resources from other queues.	Partially supported in Capacity Scheduler. This can be achieved by using the following configurations together: • yarn.resourcemanager.monitor.capacity.preemption.natural_termination_factor • yarn.resourcemanager.monitor.capacity.preemption.max_wait_before_kill Supported by Queue Manager UI.

Property	Description	Conversion information
fairSharePreemptionThreshold	The fair share preemption threshold for the queue.	Partially supported in Capacity Scheduler. This can be achieved by using the following configurations together: • yarn.resourcemanager.monitor.capacity.preemption.natural_termination_factor • yarn.resourcemanager.monitor.capacity.preemption.max_wait_before_kill Supported by Queue Manager UI.

Table 16: Placement rules

Fair Scheduler placement rules	Description	Conversion information
create="false" or "true"	Disable or enable creating a queue dynamically in YARN. This option can be specified on all rules.	Partially supported in Capacity Scheduler. Use the Capacity Scheduler Dynamic Queue Mappings policies: • u:%user:[managedParentQueueName].[queueName] • u:%user:[managedParentQueueName].%user • u:%user:[managedParentQueueName].%primary_group • u:%user:[managedParentQueueName].%secondary_group Supported by Queue Manager UI.
<rule name="specified"/>	If a user has submitted the application by specifying a queue name (other than the “default” queue), then this rule will be successful. Hence the remaining set of rules won't be executed.	Not supported in Capacity Scheduler.
<rule name="primaryGroupExistingQueue"/>	If submitted user’s(userA) primary group name (groupA) exists, submit to groupA.	There is an equivalent placement rule in Capacity Scheduler: <value>u:%user:%primary_group</value> Supported by Queue Manager UI.
<rule name="secondaryGroupExistingQueue" />	If submitted user’s(userA) secondary group name (groupA) exists, submit to groupA.	There is an equivalent placement rule in Capacity Scheduler: <value>u:%user:%secondary_group</value> Supported by Queue Manager UI.
<rule name="nestedUserQueue">	If submitted the embedded rule, all rules are allowed except for the reject rule, is executed to generate a parent queue and the user’s (userA) name is created as a child of the parent.	Not supported in Capacity Scheduler.
<rule name="default" queue=”qName”/>	Fall back policy by which rule will fall back to queue named in the property ‘queue’ or the “default“ queue if no queue property is specified (if all matches fail).	There is an equivalent placement rule in Capacity Scheduler: <value>u:%user:default</value> Supported by Queue Manager UI.

Converting Fair Scheduler to Capacity Scheduler using the fs2cs conversion utility

You can use the fs2cs conversion utility to automatically convert certain Fair Scheduler configuration to Capacity Scheduler configuration.

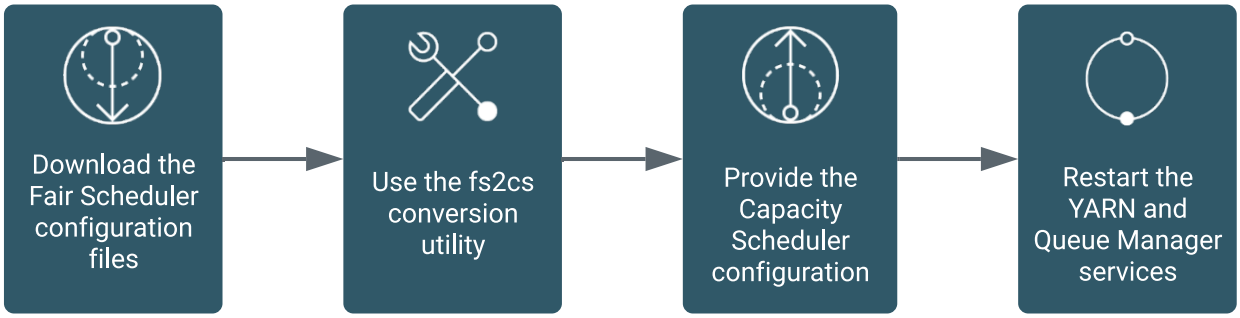
About this task

From the Cloudera Base on premises 7.1 release, Cloudera provides a conversion tool, called fs2cs conversion utility. This utility is a CLI application that is part of the yarn CLI command. It generates capacity-scheduler.xml and yarn-site.xml as output files.

From Cloudera Base on premises 7.1.6, the fs2cs conversion utility converts the scheduler configuration from weight resource allocation mode in Fair Scheduler to weight resource allocation mode in Capacity Scheduler. However, this can be changed by using the -pc command line option. If -pc is used, the fs2cs conversion utility converts from weight resource allocation mode to relative (percentage) mode.

Important: The features of Capacity Scheduler are not exactly the same as the features of Fair Scheduler. Hence, the fs2cs conversion utility cannot convert every Fair Scheduler configuration into a Capacity Scheduler configuration. After the automatic conversion and once the migration is completed, you must manually tune the scheduler configurations to ensure that the resulting scheduling configuration fits your organization's internal goals and SLAs after conversion.

When using the fs2cs utility to automatically convert your Fair Scheduler into a Capacity Scheduler, you have to perform the following steps:



Before you begin

- Be aware of the Fair Scheduler properties that are auto-converted, those that require manual configuration, and those that do not have an equivalent feature in Capacity Scheduler. To learn about how properties are converted, see Plan your scheduler migration.
- You must have downloaded and distributed parcels for the target version of Cloudera.

Procedure

1. Download the Fair Scheduler configuration files from the Cloudera Manager data store:
   a) In Cloudera Manager for the CDH cluster, navigate to Host All Hosts .
   b) Find the host with the ResourceManager role and click the YARN ResourceManager role.
   c) Click Process.
   d) Find and save the fair-scheduler.xml and yarn-site.xml configuration files for future reference.

2. Use the fs2cs conversion utility:
   a) Log in to the host machine which is part of the Cloudera cluster using ssh.
   b) Copy the downloaded XML files to this host using scp.
   c) Create a new directory to save the capacity-scheduler.xml file that is generated by the fs2cs conversion utility:

      ```
      $ mkdir -p output
      ```

   d) Use the fs2cs conversion utility to auto-convert the structure of resource pools:

      ```
      $ yarn fs2cs [--cluster-resource <VCORES/MEMORY>][--no-terminal-rule-check] --yarnsiteconfig </PATH/TO/YARN-SITE.XML> [--fsconfig </PATH/TO/FAIR-SCHEDULER.XML>] --output-directory </OUTPUT/PATH/> [--print] [--skip-verification]
      ```

   For example, if ou copied the XML files to /home/migration, then you can use the following command:

   ```
   $ yarn fs2cs --yarnsiteconfig /home/migration/yarn-site.xml --fsconfig /home/migration/fair-scheduler.xml --output-directory /home/migration/output --no-terminal-rule-check
   ```

   For a detailed list of switches supported by the fs2cs conversion utility, use yarn fs2cs --help.

   Note: You have to provide an absolute path for the yarn-site.xml and the fair-scheduler.xml configuration file. If only the file names are provided the command fails.

3. Provide the generated Capacity Scheduler configuration in Cloudera Manager:
   a) Remove the following invalid tags from the fs2cs conversion utility generated yarn-site.xml file:
      - header
      - configuration tags
      - final tags
      - source tags
   b) In Cloudera Manager for the Cloudera cluster, select the YARN service.
   c) Click Configuration.
   d) Search for capacity-scheduler and find the Capacity Scheduler Configuration Advanced Configuration Snipper (Safety Valves).
   e) Click View as XML and insert the full content of the capacity-scheduler.xml file that was generated by the converter tool.
   f) Click Save Changes.
   g) Search for yarn-site and find the YARN Service Advanced Configuration Snippet (Safety Valve) for yarn-site.xml.
   h) Click View as XML and insert the full content of the yarn-site.xml file that was generated by the converted tool.
   i) Click Save Changes.
4. Restart the YARN and Queue Manager services.

If the fs2cs conversion utility command fails, check if you provided the correct absolute path for the yarn-site.xml and the fair-scheduler.xml configuration file.

What to do next

Proceed with the migration to CDP.

After the migration is completed, manually tune the configuration generated by the fs2cs conversion utility using Queue Manager UI and Cloudera Manager Advanced configuration snippet (Safety Valves).

CLI options of the fs2cs conversion tool

List of the CLI options of the fs2cs conversion tool.

Option	Description
-d,--dry-run	Performs a dry-run of the conversion. Outputs whether the conversion is possible or not.
-f,--fsconfig <arg>	Absolute path to a valid fair-scheduler.xml configuration file. By default, yarn-site.xml contains the property which defines the path of fair-scheduler.xml. Therefore, the -f / --fsconfig settings are optional.
-h,--help	Displays the list of options.
-o,--output-directory <arg>	Output directory for yarn-site.xml and capacity-scheduler.xml files. Must have write permission for the user who is running this script. If -p or --print is specified, the xml files are emitted to the standard output, so the -o / --output-directory is ignored.
-p,--print	If defined, the converted configuration will only be emitted to the console. If -p or --print is specified, the xml files are emitted to the standard output, so the -o / --output-directory is ignored.
-pc,-percentage	This option is supported in Cloudera Base on premises 7.1.6 and higher versions. By default the fs2cs conversion utility converts into weight mode. Using -pc you can change it to relative (percentage) mode. Note that there are some scheduler transition limitations that are resolved in weight mode but not in relative (percentage) mode. Relative mode can be considered the “legacy” mode of Capacity Scheduler, where capacities are expressed in percentages.
-r,--rulesconfig <arg>	Optional parameter. If specified, should point to a valid path to the conversion rules file (property format).
-s, --skip-verification	It does not validate the converted Capacity Scheduler configuration. By default, the utility starts an internal Capacity Scheduler instance to see whether it can start up properly or not. This switch disables this behaviour.
-t,--no-terminal-rule-check	Disables checking whether a placement rule is terminal to maintain backward compatibility with configs that were made before YARN-8967. By default, Fair Scheduler performs a strict check of whether a placement rule is terminal or not. This means that if you use a <reject> rule which is followed by a <specified> rule, then this is not allowed, since the latter is unreachable. However, before YARN-8967, Fair Scheduler was more lenient and allowed certain sequences of rules that are no longer valid. Inside the tool, a Fair Scheduler instance is instantiated to read and parse the allocation file. In order to have Fair Scheduler accept such configurations, the -t or --no-terminal-rule-check argument must be supplied to avoid the Fair Scheduler instance throwing an exception.
-y,--yarnsiteconfig <arg>	Path to a valid yarn-site.xml configuration file.

Manual configuration of scheduler properties

After migrating to Cloudera, you must manually fine-tune the scheduler configurations using the YARN Queue Manager UI to ensure that the resulting configurations suit your requirements. You can use Cloudera Manager Advanced configuration snippet (Safety Valve) to configure a property that is missing from the YARN Queue Manager UI.

The features of Capacity Scheduler are not exactly the same as the features of Fair Scheduler. Hence, the conversion utility cannot convert every Fair Scheduler configuration into a Capacity Scheduler configuration. Therefore, you must manually tune the scheduler configurations to ensure that the resulting scheduling configuration fits your organization's internal goals and SLAs after conversion. If needed, further change the scheduler properties in the capa city-scheduler.xml and yarn-site.xml output files generated by the fs2cs conversion utility. For information about the Fair Scheduler properties that are auto-converted by the fs2csconversion utility, see *Auto-converted Fair Scheduler properties*.

You can configure the properties manually using the YARN Queue Manager UI. If you see a property that is unavailable in the Queue Manager UI, you can use Cloudera Manager configuration snipper (Safety Valves) to configure them.

Important: You must not use the Queue Manager UI and Cloudera Manager Safety Valves at the same time as safety valves overwrite the configuration set using Queue Manager UI.

Related Information

Auto-converted Fair Scheduler properties

Managing and allocating cluster resources using Capacity Scheduler

Using YARN Queue Manager UI to configure scheduler properties

After migrating to Cloudera, you must configure the Capacity Scheduler properties using the output files generated by the fs2cs conversion utility. You can configure the properties manually using the YARN Queue Manager UI service.

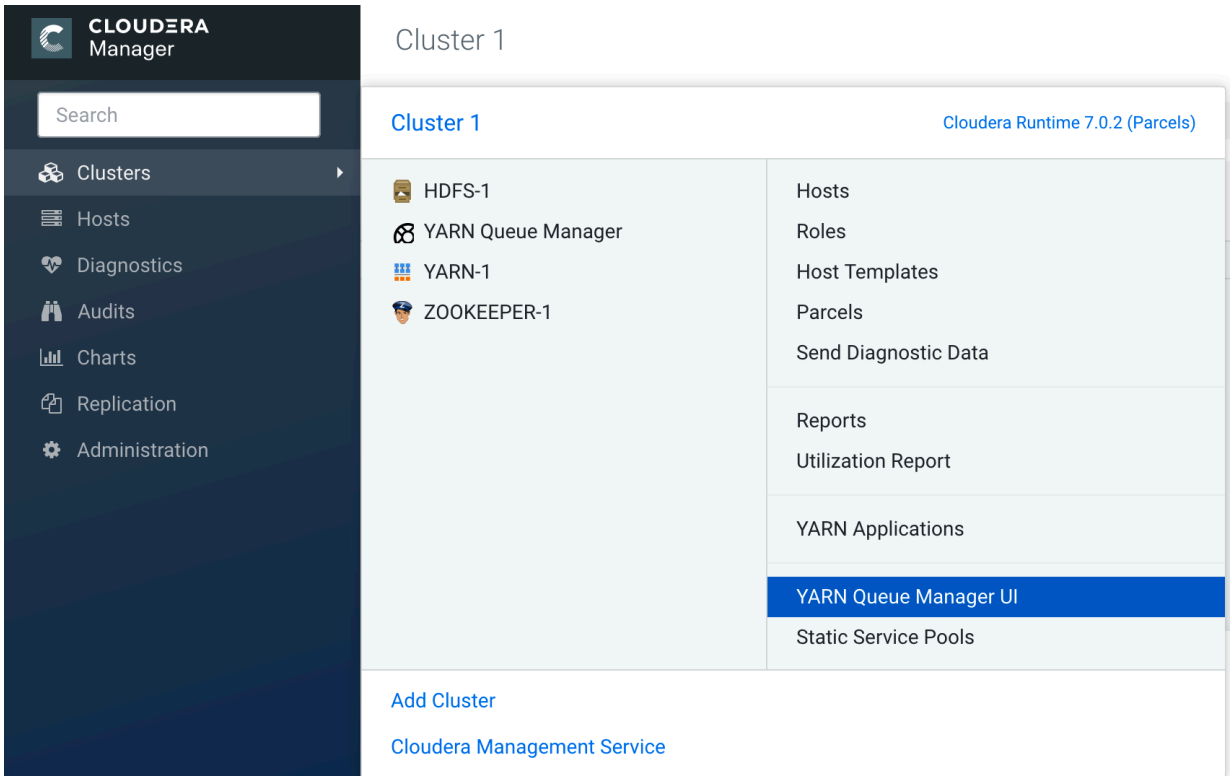
Before you begin

- Use the fs2cs conversion utility to generate the capacity-scheduler.xml and yarn-site.xml output files.
- Complete the scheduler migration process.
- Identify properties that require manual configuration and can be configured using the Queue Manager UI.

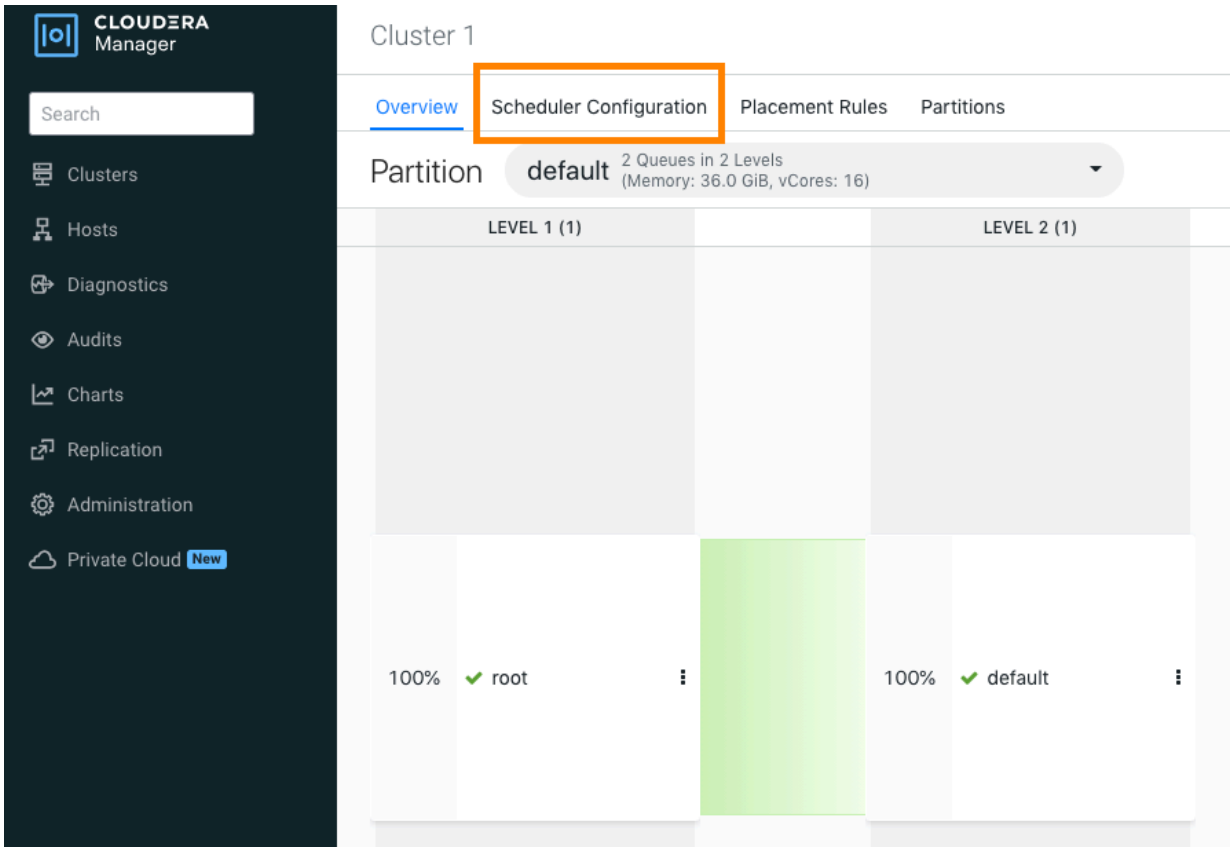
For more information about scheduler properties, see *Fair Scheduler feature and conversion details*.

Procedure

1. In Cloudera Manager, click Clusters and select the YARN Queue Manager UI service.



2. In the YARN Queue Manager window, click the Scheduler Configuration tab.



3. In the Scheduler Configuration window, enter the value of the property and click Save.

Related Information

Fair Scheduler features and conversion details

Using Cloudera Manager Safety Valves to configure scheduler properties

Certain scheduler properties can neither be converted by the fs2cs conversion utility nor be configured using the YARN Queue Manager UI service. After migrating to Cloudera, you must manually configure these properties using the Cloudera Manager advanced configuration snippet (Safety Valves).

Before you begin

- Use the fs2cs conversion utility to generate the capacity-scheduler.xml and yarn-site.xml output files.
- Complete the scheduler migration process.
- Identify the scheduler properties that need to be configured manually and not supported by the Queue Manager UI. For more information, see Fair Scheduler features and conversion details.

Procedure

1. In Cloudera Manager, select the YARN service.
2. Click the Configuration tab.
3. Search for capacity-scheduler, and find the Capacity Scheduler Configuration Advanced Configuration Snippet (Safety Valve).
4. Click View as XML, and insert the complete capacity-scheduler.xml file, generated by the converter tool.
5. Add the necessary configuration properties.
6. Click Save Changes.
7. Search for yarn-site, and find the YARN Service Advanced Configuration Snippet (Safety Valve) for yarn-site.xml.
8. Click View as XML and add the required configuration in an XML format.

   Optionally, use + and - to add and remove properties.
9. Click Save Changes.
10. Restart the YARN service.