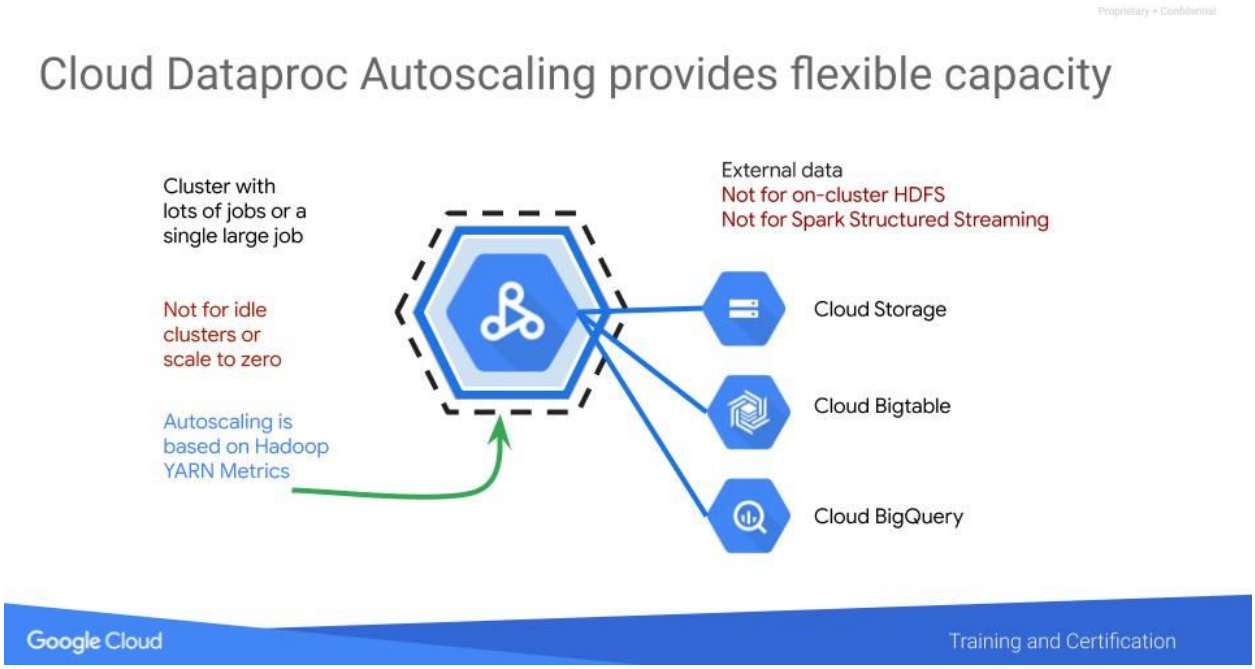
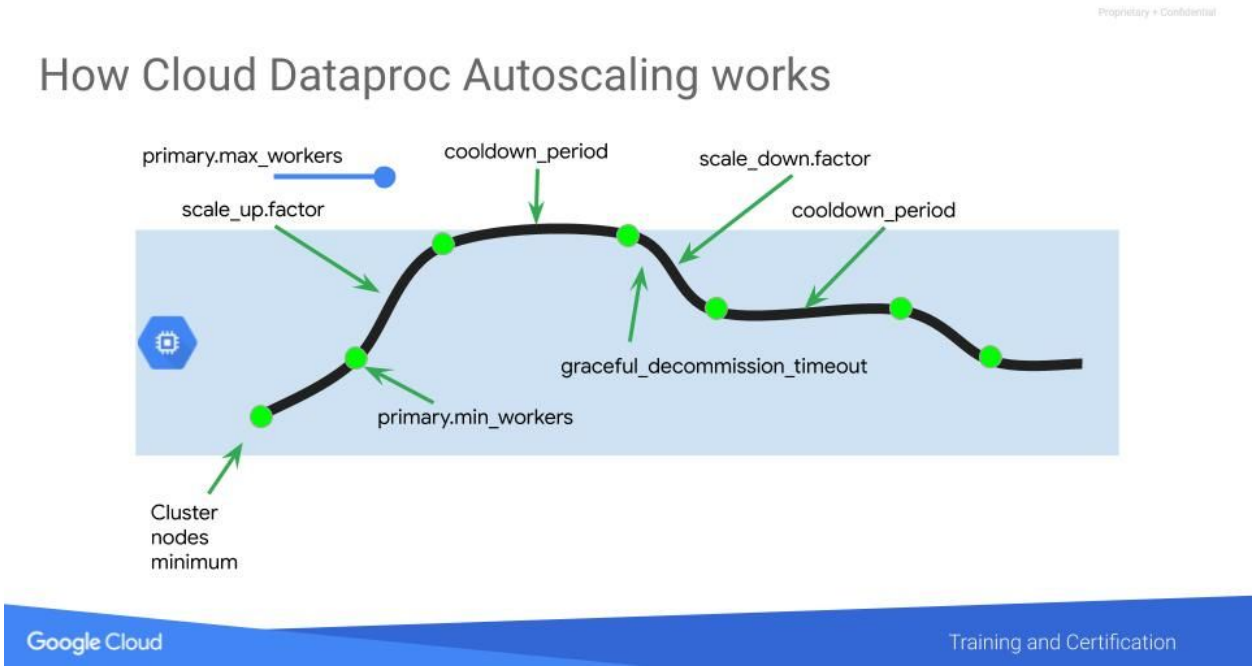
Cloud Dataproc Automation Features
Autoscaling and Cluster Deletion



Cloud Dataproc Autoscaling provides flexible capacity for more efficient utilization.
It makes scaling decisions based on Hadoop YARN Metrics.
It is designed to be used only with off-cluster persistent data, not on-cluster HDFS or HBase.
It works best with a cluster that processes a lot of jobs or that processes a single large job.

It does not support Spark Structured Streaming (a streaming service built on top of Spark SQL).
It is not designed to scale to zero. So it is not the best for sparsely utilized or idle clusters.
In these cases it is equally fast to terminate a cluster that is idle and create a new cluster when it is needed.
For that purpose you would look at Cloud Dataproc Workflows or Cloud Composer, and Cluster Scheduled Deletion.



Initial workers -- The number of initial workers is set from Worker Nodes > nodes minimum. Setting this value ensures that the cluster comes up to basic capacity faster than if you let autoscaling handle it. Because autoscaling might require multiple autoscale periods to scale up.

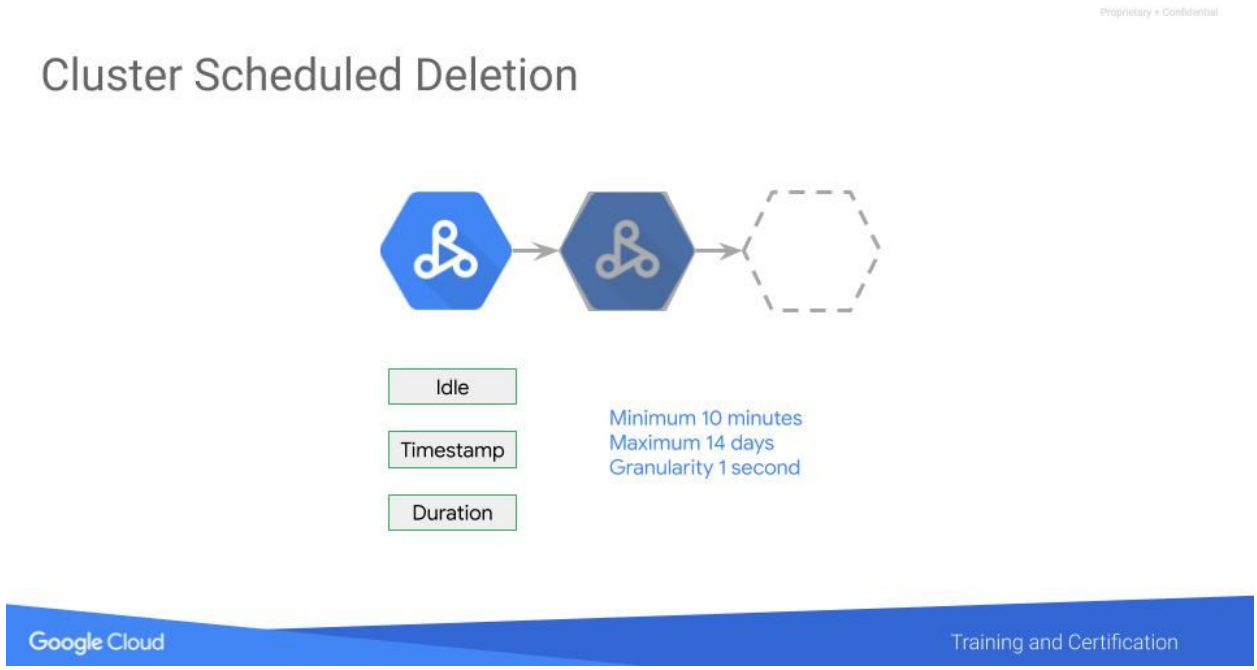
The primary minimum number of workers may be the same as the cluster nodes minimum. There is a maximum that caps the number of worker nodes. Now there is heavy load on the cluster. And Autoscaling determines it is time to scale up. The scale_up.factor determines how many nodes to launch. This would commonly be one node. But if you knew that a lot of demand would occur at once, maybe you want to scale up faster. After the action, there is a cooldown period to let things settle before autoscaling evaluation occurs again. The cooldown period reduces the chances that the cluster will start and terminate nodes at the same time.

In this example, the extra capacity isn't needed. And there is a graceful decommission timeout to give running jobs a chance to complete before the node goes out of service. Notice there is a scale down factor. In this case it is scaling down by one node at a time for a more leisurely reduction in capacity.

After the action, there is another cooldown period.
And a second scale down, resulting in a return to the minimum number of workers.
A secondary min_workers and max_workers controls the scale of preemptible workers.

You can read about the scaling algorithm here:
https://cloud.google.com/dataproc/docs/concepts/configuring-clusters/autoscaling

- scale_up.factor -- how many nodes to add during a scale-up event.
- scale_down.factor -- how many nodes to remove during a scale-down.event.
- graceful_decomission_timeout -- how long to wait for a jobs to complete before shutting down the node.



Efficient utilization, don't pay for resources you don't use.

A fixed amount of time after the cluster enters the Idle state.
Set a timer. You give it a timestamp. The count starts immediately once the expiration has been set.
Set a duration. Time in seconds to wait before deleting the cluster.
Range is from 10 minutes minimum to 14 days maximum, with a granularity of 1 second.

Currently available from the command line and REST API, but not through Console.

https://cloud.google.com/dataproc/docs/concepts/configuring-clusters/scheduled-deletion

RFC 3339 UTC "Zulu" format, accurate to nanoseconds
Provide a duration in seconds with up to nine fractional digits, terminated by 's'. Example: "3.5s".