**Troubleshooting Kubernetes Worker Node notReady**

2018-05-06

***(Last Updated On: 2018-05-06)***

The solutions in this post are geared toward those who are hosting their cluster with AWS and managing the cluster with KOPS.  However, the troubleshooting steps apply the most scenarios.  I am sharing this in the hopes of saving others the stress that I experienced the first time this happened to me.  If you have not had a Kubernetes worker node go in to notReady state, read on because you will.

If you are here because you have a worker node in notReady state right now and you are using AWS and KOPS, follow the troubleshooting steps below.  I will discuss them afterwards.

1. Run kubectl get nodes to get the name of the nodes in notReady state.
   * At this point I recommend provisioning additional nodes with KOPS to relieve pressure on your other nodes and give the pods on the the notReady nodes a place to go.
     1. kops edit ig nodes to bring up the editor.
     2. Set the maxSize and minSize values.
     3. Preview the changes with kops update cluster <clustername>.
     4. Apply the changes with kops update cluster <clustername> --yes.
   * Detailed instructions are available [here](https://github.com/kubernetes/kops/blob/master/docs/instance_groups.md).
2. Use kubectl describe node <node name> to get the status of the node.
   * A handy shortcut to the two steps above is kubectl get nodes | grep '^.\*notReay.\*$' | awk '{print $1}' | xargs kubectl describe node
3. Look for the Conditions heading and check the condition of NetworkUnavailable, OutOfDisk, MemoryPressure, and DiskPressure .
   * If the statuses of those items are helpful, begin troubleshooting those conditions.
   * If there are memory or disk issues, there is a good chance that you have a pod or a number of pods wreaking havoc.  Fix the problems with those pods if you can, otherwise, prevent them from being scheduled to other nodes by deleting them, or using kubectl cordon <node name> on your healthy nodes to prevent new pods from being scheduled to them.
4. SSH in to the unhealthy nodes.  If you cannot SSH into the nodes, skip ahead.  Otherwise, use ps -eaf to determine if the docker daemon and kublet are running.
5. If you have determined that kubelet and the docker daemon are not running on the node, or you are are not able to determine this, use the AWS console or AWS CLI to terminate the node.  The autoscale group created by KOPS will provision a new node.
   * At this time, if you provisioned additional nodes that you would like to remove, use kubectl drain <node-name> to drain the node.  Once it has been drained, update the cluster configuration with KOPS to reduce the size of the cluster, and then terminate the node.

Of course, if you aren’t using KOPS and an autoscale group, these steps won’t be as helpful to you.  In general, I think it is important to quickly diagnose whether or not the docker daemon and kubelet are running on the affected nodes.  If the docker daemon is down and cannot be restarted, a quick and simple solution might be replacing the node, assuming you can do that quickly and safely.  After all, once the docker daemon is down, your pods aren’t working anyway.

If you have not had the need to fix a node in notReady state, **now** is a good time plan what you will do when you encounter this situation.  For my [hobby cluster](https://joecreager.com/cheaper-pastures/) that I host with [Linod](https://www.linode.com/?r=fd8e75d10f90dc1332d1c250acf77b5aa6904d67), I will do roughly the same procedure that I describe above, except I won’t be able to terminate the node and expect it to be replace automatically.  Instead, I will use kube-linode to provision an additional node to transfer the orphaned pods to.  Then, I will use kubectl delete <node name> to remove the node object, and finally remove the node from my Linode account.

Unfortunately, the steps above do not identify what caused the node to go in to notReady state.  Additionally, once the node is gone, you will probably lose any logs that you might have been able to snoop to figure out what happened.  Again, **now** is a good time to plan what how you will deal with this situation when it comes up.  I recommend monitoring your nodes with node\_exporter and [Prometheus](https://joecreager.com/deploy-prometheus-operator/), and using some type of log aggregation tool.  At work, I use LogDNA.