Debugging Your Kubernetes Nodes in the ‘Not Ready’ State | nodenotready

Kubernetes clusters typically run on multiple “nodes” each having its own state. In this article, you’ll learn a few possible reasons a node might enter the NotReady state and how you can debug it.

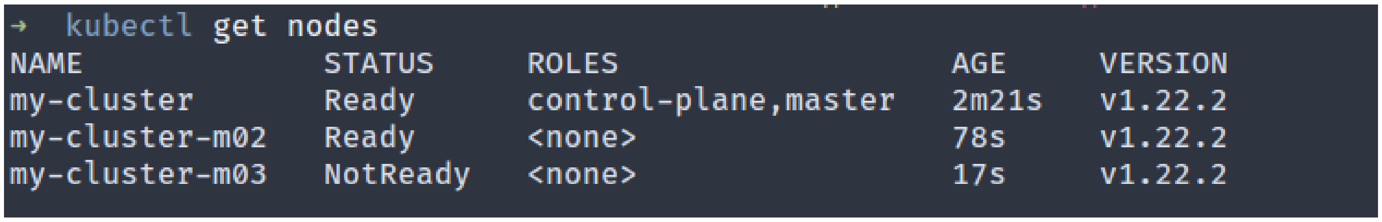
Aniket Bhattacharyea

Software Engineer

Overview

Nodes are a vital component of a Kubernetes cluster and are responsible for running the pods. Depending on your cluster setup, a node can be a physical or a virtual machine. A cluster typically has one or multiple nodes, which are managed by the control plane.

Because nodes do the heavy lifting of managing the workload, you want to make sure all your nodes are running correctly. The `kubectl get nodes` command can be used to check the state of your nodes.



```
→  kubectl get nodes
NAME              STATUS     ROLES                  AGE     VERSION
my-cluster        Ready      control-plane,master   2m21s   v1.22.2
my-cluster-m02    Ready      <none>                 78s     v1.22.2
my-cluster-m03    NotReady   <none>                 17s     v1.22.2
```

Output of kubectl get nodes

A node with a `NotReady` status means it can’t be used to run a pod because of an underlying issue. It’s essentially used to debug a node in the `NotReady` state so that it doesn’t lie unused.

In this article, you’ll learn a few possible reasons why a node might enter the `NotReady` state and how you can debug it.

The NotReady State

As mentioned earlier, each node in a cluster is used to run pods. Before a pod is scheduled on a node, Kubernetes checks whether the node is capable of running the pod or not. The `STATUS` column in the output of `kubectl get nodes` represents the status. The possible values in this column are:

1. `Ready`: The node is healthy and ready to accept pods.
2. `NotReady`: The node has encountered some issue and a pod cannot be scheduled on it.
3. `SchedulingDisabled`: The node is marked as unschedulable. This can be done using the kubectl cordon command.
4. `Unknown`: The node is unreachable by the control plane.

Having a node in the `NotReady` state means that the node is effectively unused and will accumulate costs without participating in running pods. Furthermore, losing a node can negatively impact your production workload.

In order for your application to run smoothly, you must debug them quickly.

Possible Causes of the NotReady State

There can be various reasons why a node might enter the `NotReady` state. This section will review some of the most common reasons for this error.

Scarcity of Resources

To operate normally, a node must have sufficient disk space, memory, and sufficient processing ability. If a node is running low on disk space or the available memory is low, it will go into the `NotReady` state. If pressure exists on the processes, *eg* too many processes are running on the node, it will also change to the `NotReady` state.

Network Misconfiguration

If the network has not been correctly configured on the node or it can't reach the internet, the node will be unable to communicate with the master node and will be listed as `NotReady`.

Issue with kubelet Process

kubelet is an agent that runs on each node. It is responsible for communicating with the Kubernetes API server and registering the nodes. If kubelet crashes or stops on the node, it will not be able to communicate with the API Server and will be in the `NotReady` state.

Issue with kube-proxy

kube-proxy is a network proxy that runs on each node and maintains the network rules. These rules allow network communication to your pods from inside or outside your cluster. If kube-proxy crashes or stops, the node will be in the `NotReady` state.

Vendor Specific Issues

Suppose you're using a cloud-hosted solution like GKE or EKS. In that case, some vendor-specific issues may be preventing your nodes from operating normally and communicating with the control plane. These issues could be IAM misconfiguration, misconfigured network rules, etc.

K8s Metrics, Logging, and Tracing

Monitor the health of your cluster and troubleshoot issues faster with pre-built dashboards that just work.

Debugging the NotReady State

As you can see, the `NotReady` status can be caused by a multitude of issues. This section will help you *identify* the root cause of the problem. However, it's essential to understand that how you go about fixing these issues depends on the exact cause and your cluster setup. There are no one-size-fits-all solutions. But, once you identify the root cause, it should be easier to resolve it.

Check the kube-proxy Pod

First, ensure that each node has exactly one `kube-proxy` pod and is in the `Running` state.

```
kubectl get pods -n kube-system -o wide
```

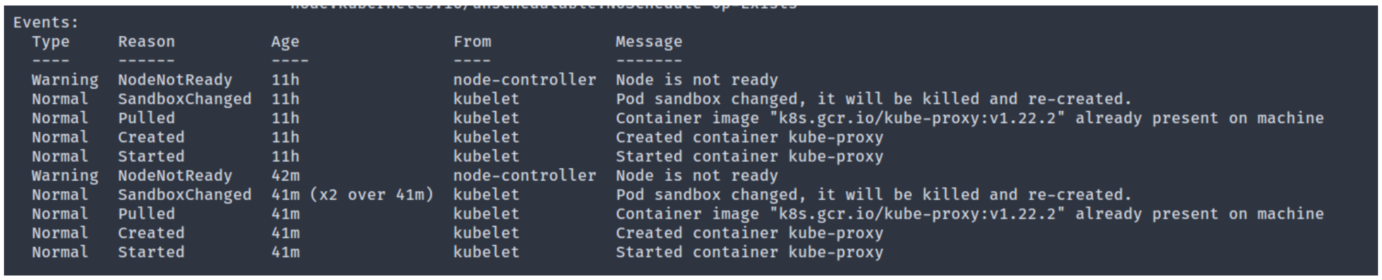
The output might look like this:

```
NAMEREADYSTATUSAGEIPNODENOMINATED NODEREADINESS GATESkube-proxy-nhbtp1
/1Running2 (11h ago)2d16h192.168.99.101 my-cluster<none><none>kube-
proxy-tkmsk1/1Running2 (11h ago)2d16h192.168.99.103 my-cluster-
m03<none><none>kube-proxy-vk4ch1/1Running2 (11h ago)2d16h192.168.99.102
my-cluster-m02<none><none>
```

If any one pod is in some state other than `Running`, use the following command to get more information:

```
kubectl describe pod yourPodName -n kube-system
```

The Events section logs the various events on the pod, and it could be an excellent place to start looking for any mishaps.



```
Events:
  Type     Reason          Age                From             Message
  ----     ------          ----               ----             -------
  Warning  NodeNotReady    11h                node-controller  Node is not ready
  Normal   SandboxChanged  11h                kubelet          Pod sandbox changed, it will be killed and re-created.
  Normal   Pulled          11h                kubelet          Container image "k8s.gcr.io/kube-proxy:v1.22.2" already present on machine
  Normal   Created         11h                kubelet          Created container kube-proxy
  Normal   Started         11h                kubelet          Started container kube-proxy
  Warning  NodeNotReady    42m                node-controller  Node is not ready
  Normal   SandboxChanged  41m (x2 over 41m)  kubelet          Pod sandbox changed, it will be killed and re-created.
  Normal   Pulled          41m                kubelet          Container image "k8s.gcr.io/kube-proxy:v1.22.2" already present on machine
  Normal   Created         41m                kubelet          Created container kube-proxy
  Normal   Started         41m                kubelet          Started container kube-proxy
```

The events section in the output

You can get access to the pod logs by running the following command:

```
kubectl logs yourPodName -n kube-system
```

The logs and the events list is a good place to start looking for any issues.

If your node does not have a `kube-proxy` pod, then you need to inspect the `kube-proxy` daemonset, which is responsible for running one kube-proxy pod on each node.

```
kubectl describe daemonset kube-proxy -n kube-system
```

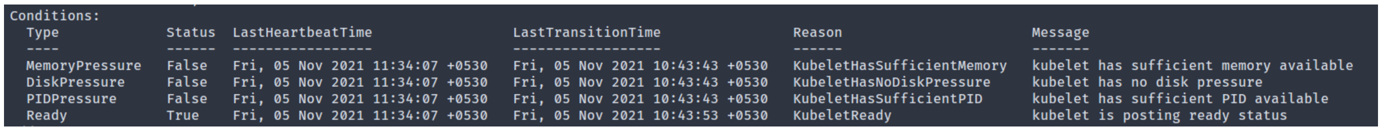
The output of this command might reveal any possible issue with the daemonset.

Verify Resources are Available

Run the following command to get detailed information about a node that is not ready:

```
kubectl describe node nodeName
```

In the output, the `Conditions` section shows if the node is running out of resources or not.



```
Conditions:
  Type             Status  LastHeartbeatTime                 LastTransitionTime                Reason                       Message
  ----             ------  -----------------                 ------------------                ------                       -------
  MemoryPressure   False   Fri, 05 Nov 2021 11:34:07 +0530   Fri, 05 Nov 2021 10:43:43 +0530   KubeletHasSufficientMemory   kubelet has sufficient memory available
  DiskPressure     False   Fri, 05 Nov 2021 11:34:07 +0530   Fri, 05 Nov 2021 10:43:43 +0530   KubeletHasNoDiskPressure     kubelet has no disk pressure
  PIDPressure      False   Fri, 05 Nov 2021 11:34:07 +0530   Fri, 05 Nov 2021 10:43:43 +0530   KubeletHasSufficientPID      kubelet has sufficient PID available
  Ready            True    Fri, 05 Nov 2021 11:34:07 +0530   Fri, 05 Nov 2021 10:43:53 +0530   KubeletReady                 kubelet is posting ready status
```

The conditions section in the output

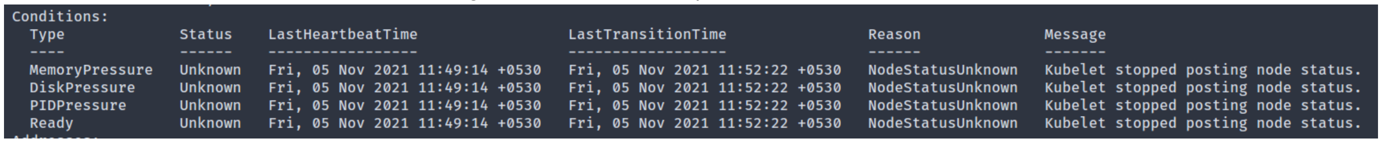
The following conditions are available:

1. `MemoryPressure`: If `True`, it indicates that the node is running out of memory.
2. `DiskPressure`: A `True` value in this field indicates that the node lacks enough space.
3. `PIDPressure`: If too many processes are running on the node, this field will be `True`.
4. `NetworkUnavailable`: If the network for the node is not correctly configured, this will be `True`.
5. `Ready`: If the node is healthy and ready to accept pods, this will be `True`. In this field, a `False` is equivalent to the `NotReady` status in the `get nodes` output. It can also have the `Unknown` value, which means the node controller has not heard from the node in the last `node-monitor-grace-period` (defaults to 40 seconds).

If any one of the first four conditions is `True`, you have identified the problem.

Verify kubelet is Running

If all the `Conditions` fields show `Unknown`, it might hint that the kubelet process on the node has run into some issues.



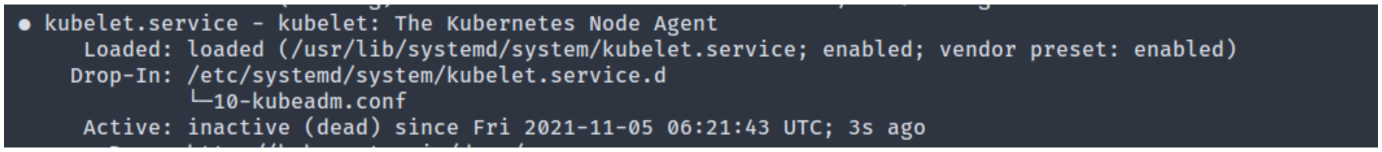
```
Conditions:
  Type             Status    LastHeartbeatTime                 LastTransitionTime                Reason              Message
  ----             ------    -----------------                 ------------------                ------              -------
  MemoryPressure   Unknown   Fri, 05 Nov 2021 11:49:14 +0530   Fri, 05 Nov 2021 11:52:22 +0530   NodeStatusUnknown   Kubelet stopped posting node status.
  DiskPressure     Unknown   Fri, 05 Nov 2021 11:49:14 +0530   Fri, 05 Nov 2021 11:52:22 +0530   NodeStatusUnknown   Kubelet stopped posting node status.
  PIDPressure      Unknown   Fri, 05 Nov 2021 11:49:14 +0530   Fri, 05 Nov 2021 11:52:22 +0530   NodeStatusUnknown   Kubelet stopped posting node status.
  Ready            Unknown   Fri, 05 Nov 2021 11:49:14 +0530   Fri, 05 Nov 2021 11:52:22 +0530   NodeStatusUnknown   Kubelet stopped posting node status.
```

The conditions field shows unknown

To debug this, first SSH into the node and check the status of the kubelet process. If it's running as a systemd service, use the following command:

```
systemctl status kubelet
```

If the `Active` field shows `inactive (dead)`, it means the kubelet process has stopped.



```
● kubelet.service - kubelet: The Kubernetes Node Agent
     Loaded: loaded (/usr/lib/systemd/system/kubelet.service; enabled; vendor preset: enabled)
    Drop-In: /etc/systemd/system/kubelet.service.d
             └─10-kubeadm.conf
     Active: inactive (dead) since Fri 2021-11-05 06:21:43 UTC; 3s ago
```

The active field of the output

To reveal the possible reason for the crash, check the logs with the following command:

```
journalctl -u kubelet
```

Once the issue is fixed, restart kubelet with:

```
systemctl restart kubelet
```

Verify Network Communication with the Control Plane

If the `Conditions` field shows `NetworkUnavailable`, it indicates an issue in the network communication between the node and the control plane.

A few possible fixes:

- If the node is configured to use a proxy, verify that the proxy allows access to the API server endpoints.
- Ensure that the route tables are appropriately configured to avoid blocking communication with the API server.
- If you're using a cloud provider like AWS, verify that no VPC network rules block communication between the control plane and the node.

You can run the following command from within the node to verify that it can reach the API server.

```
nc -vz <your-api-server-endpoint> 443
```

If the output shows `succeeded`, then network communication is working correctly.

Vendor Specific Debugging

If you're using a cloud provider like EKS, or GKE, sometimes it's worth looking into vendor-specific issues if you've exhausted all other debugging techniques. EKS has an extremely detailed guide that you can follow.

GKE provides an auto repair feature that can attempt to repair a node that has been in the `NotReady` state for a given amount of time. If all else fails, you can always get in touch with your cloud provider for more assistance.

Final Thoughts

Having a node in the `NotReady` state is undesirable and needs to be fixed immediately. However, there are multiple reasons this might occur, and it can be challenging to pinpoint the exact cause. This article discussed some common reasons you may encounter the `NotReady` command and solutions for it.

The earlier you can catch nodes entering the `NotReady` state, the higher your chances of quickly debugging it. ContainIQ is a Kubernetes monitoring platform that comes with an extensive events dashboard that can monitor and alert you when a node enters the `NotReady` state. You can also see all the events leading up to this, allowing you to quickly identify and solve the issue.