

Troubleshooting Kubernetes

Introduction

1. It has two sections
 1. [Troubleshooting your application](#) - useful for users who are deploying code into Kubernetes and wondering why it is not working
 2. [Troubleshooting your cluster](#) - useful for cluster administrators and people whose Kubernetes cluster is unhappy

Getting help

1. If problems weren't answered by the guides, there are a variety of ways for us to get help from Kubernetes community

Questions

1. Documentation on the site has been structured to provide answers to a range of questions
2. [Concepts](#) - explains Kubernetes architecture and how each component works
3. [Setup](#) - provides practical instructions for getting started
4. [Tasks](#) - Show how to accomplish commonly used tasks
5. [Tutorials](#) - comprehensive walkthroughs of real-world, industry-specific, or end-to-end development scenarios
6. [Reference](#) - provides detailed documentation on [Kubernetes API](#) and command-line interfaces (CLIs) such as [kubect!](#)

Help! My question isn't covered! I need help now!

Stack Overflow

1. Kubernetes team also monitors [posts tagged Kubernetes](#)
2. We can [ask a new one!](#)

Slack

1. `#kubernetes-users` channel
 1. We can [request and invitation](#)
 2. Registration is open for everyone

3. We can ask any questions
4. We can access [Kubernetes organisation in Slack](#) via web browser or Slack app
5. We can browse channels for various subjects of interest
 1. `#kubernetes-novice` - for people new to Kubernetes [#kubernetes-novice](#)
 2. `#kubernetes-dev` - for developers [#kubernetes-dev](#)
 3. Country specific or local language specific channels
 1. `#cn-users` , `#cn-events`
 2. ...
 3. `#in-users` , `#in-events`

Forum

1. [discuss.kubernetes.io](#) - Official Kubernetes forum

Bugs and feature requests

1. [Github issue tracking system](#) - If we possibly discover a bug, we can make a feature request
 1. See existing issues to see if issue is already covered
 2. We can include detailed information about how to reproduce the problem
 1. Kubernetes version: `kubectl version`
 2. Cloud provider, OS distro, network configuration, and Docker version
 3. Steps to reproduce the problem

Troubleshoot Applications

1. To help users debug applications deployed into Kubernetes and not behaving correctly

Diagnosing the Problem

Debugging Pods

1. Check current state of the Pod and recent events:

```
kubectl describe pods ${POD_NAME}
```

1. Check the status of containers in the pod
 1. Are all of them **Running**?
 2. Have there been any recent restarts
 3. Debug based on the state of the pods

My pod stays pending

1. If Pod is stuck in **Pending** - it cannot be scheduled onto a node
 1. Usually because there are insufficient resources of one type or another that prevents scheduling
 1. `kubectl describe ...` has messages from scheduler about why it can not schedule a pod
 1. Reasons can be:
 1. **No enough resources** - exhausted CPU or Memory in cluster
 1. Solution:
 1. Delete pods
 2. Adjust resource requests OR add new nodes to cluster [Compute resources documentation](#)
 2. **Using hostPort** - Binding Pod to **hostPort** can result in there being limited number of places that pod can be scheduled
 1. It is unnecessary in most cases
 2. Solution: Use Service object to expose the pod instead
 3. Solution: If **hostPort** must be used, the number of pods must match the number of nodes in the cluster

My pod stays waiting

1. If a Pod is stuck in **Waiting** state
 1. It has been scheduled to a worker node
 2. But it cannot run on that machine
 3. Diagnosis: `kubectl describe ...`
 4. Common cause(s):
 1. Failure to pull the image
 2. Diagnosis:
 1. Ensure that the name of the image is correct
 2. We have the image pushed to the repository
 3. Run `docker pull <image>` to see if the image can be pulled

My pod is crashing or otherwise unhealthy

1. [Debug Running Pods](#) - once pods is scheduled

My pod is running but not doing what I told it to do

1. There could be an error in pod description (`mypod.yaml` in local machine) & error was ignored

silently when pod was created

1. Check for nesting issues in the sections
2. Check for key names (key might have been ignored)
 1. `command` if misspelled as `commnd` - pod gets created but will not use the commandline we intended to be used

2. Possible solution:

1. Delete the pod and try creation again with `--validate` option

```
kubectl apply --validate -f mypod.yaml
```

1. It gives an error if `command` is misspelled as `commnd`

```
I0805 10:43:25.129850 46757 schema.go:126] unknown field: commnd
```

2. Check if the pod is on the apiserver machine that we meant it to be on

```
kubectl get pods/mypod -o yaml > mypod-on-apiserver.yaml
```

1. Compare the description with local pod description (`mypod.yaml` say)

```
# The following must not show lines that are not in the original file m  
ypod.yaml  
diff mypod.yaml mypod-on-apiserver.yaml
```

1. There will be some lines on "apiserver" version that are not on original version (expected)
2. If we don't have lines in apiserver that are in the original version, then it might indicate a problem with pod spec

Debugging Replication Controllers

1. Replication controllers are straight forward
 1. They can either instantiate Pods or not
 1. If they can't then debug using instruction previously discussed
2. The following command can be used to introspect events related to replication controller

```
kubectl describe rc ${CONTROLLER_NAME}
```

Debugging Services

1. Services provide load balancing across a set of pods
2. There could be several problems that can make Services not work properly

1. Debugging Service problems

1. Verify that there are endpoints for service (For every service object, apiserver makes an **endpoints** resource available)

```
kubectl get endpoints ${SERVICE_NAME}
```

1. Ensure that endpoints match with the number of pods that we expect to be members of the service
 1. Example: If Service has 3 replicas, we must see 3 IP addresses in Service's endpoints

My service is missing endpoints

1. If missing endpoints, we can try listing pods using labels that Services uses

1. Consider the following service

```
...
spec:
  - selector:
      name: nginx
      type: frontend
```

2. Run the following command:

```
kubectl get pods --selector=name=nginx,type=frontend
```

1. Lists pods that match the selector
2. Check if list matches Pods that we expect to provide our Service
3. Verify that pod's **containerPort** matches up with Service's **targetPort**

Network traffic is not forwarded

1. [debugging services](#)

What's next

1. If none of the above solves the problem
 1. [Debugging Service document](#) - To ensure that

1. **Service** is running
2. has **Endpoints**
3. **Pods** are actually serving
4. has DNS working
5. iptables rules installed
6. kube-proxy is behaving as expected

2. [troubleshooting document](#) - more info

Debugging Running Pods

1. [Debugging Running Pods](#)