

# How-to validate the Kubernetes manifests

Datree allows developers to verify the Kubernetes configuration before applying it directly into the Production, Stage, Test, or even the Development environment. Datree is a command-line tool that is installed locally onto the developer's machine so that he or she can run the CLI commands that verify the Kubernetes manifests (YAML files).

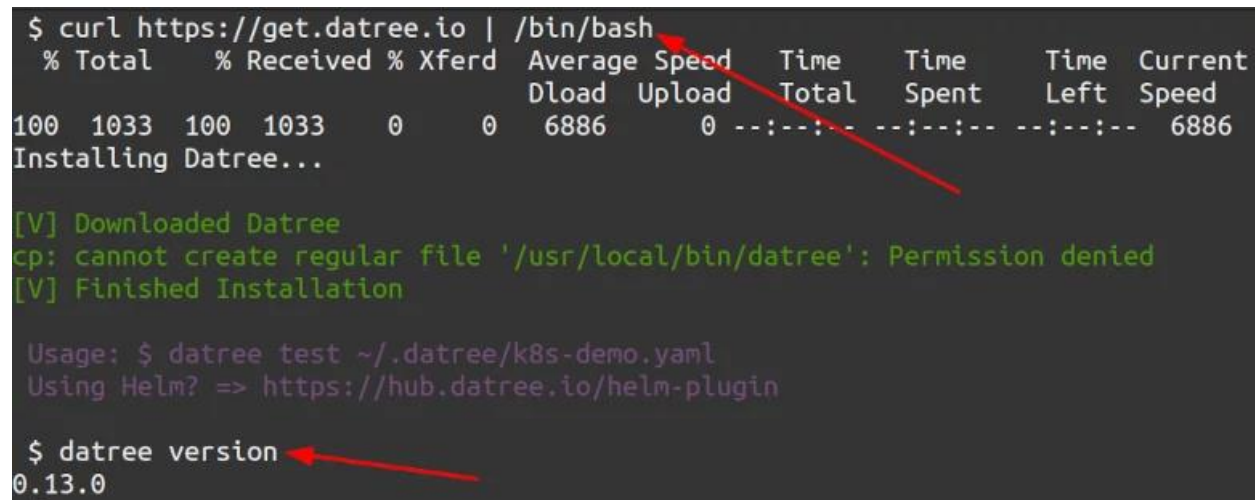
Here is a short command snippet of Datree -

```
# datree test my-kubernetes-manifest.yaml
```

On Linux, you can simply install it by running the following command -

**Linux**

```
# curl https://get.datree.io | /bin/bash
```



```
$ curl https://get.datree.io | /bin/bash
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           %             0      0     6886           0 --:--:-- --:--:-- --:--:-- 6886
Installing Datree...

[V] Downloaded Datree
cp: cannot create regular file '/usr/local/bin/datree': Permission denied
[V] Finished Installation

Usage: $ datree test ~/.datree/k8s-demo.yaml
Using Helm? => https://hub.datree.io/helm-plugin

$ datree version
0.13.0
```

## How to test Kubernetes Manifest using Datree

After installing the Datree, the next step is to test the Kubernetes manifest (YAML). To do this, you must have a copy of your Kubernetes manifest (YAML) available locally where you have already installed Datree.

```
# touch k8s-spring-boot-deployment.yaml
```

```
# nano k8s-spring-boot-deployment.yaml
```

Input this information into the Kubernetes yaml file

```
apiVersion: apps/v1
```

kind: Deployment

metadata:

name: jhooq-springboot

spec:

replicas: 3

selector:

matchLabels:

app: jhooq-springboot

template:

metadata:

labels:

app: jhooq-springboot

spec:

containers:

- name: springboot

image: rahulwagh17/jhooq-docker-demo:jhooq-docker-demo

ports:

- containerPort: 8080

env:

- name: PORT

value: "8080"

---

apiVersion: v1

kind: Service

metadata:

```
name: jhooq-springboot
```

```
spec:
```

```
type: NodePort
```

```
ports:
```

```
- port: 80
```

```
targetPort: 8080
```

```
selector:
```

```
app: jhooq-springboot
```

```
ctrl + x
```

```
yes
```

```
enter
```

Here is a command to test my Kubernetes manifest

Here is a command to test my Kubernetes manifest -

```
# datree test k8s-spring-boot-deployment.yml
```

When you run the above command, you should see the following validation error messages -

```

$ datree test k8s-spring-boot-deployment.yml
>> File: k8s-spring-boot-deployment.yml

[V] YAML validation
[V] Kubernetes schema validation
[X] Policy check

✗ Ensure each container has a configured memory request [1 occurrence]
  - metadata.name: jhooq-springboot (kind: Deployment)
  ⚡ Missing property object 'requests.memory' - value should be within the accepted boundaries recommended by the organization

✗ Ensure each container has a configured CPU request [1 occurrence]
  - metadata.name: jhooq-springboot (kind: Deployment)
  ⚡ Missing property object 'requests.cpu' - value should be within the accepted boundaries recommended by the organization

✗ Ensure each container has a configured memory limit [1 occurrence]
  - metadata.name: jhooq-springboot (kind: Deployment)
  ⚡ Missing property object 'limits.memory' - value should be within the accepted boundaries recommended by the organization

✗ Ensure each container has a configured CPU limit [1 occurrence]
  - metadata.name: jhooq-springboot (kind: Deployment)
  ⚡ Missing property object 'limits.cpu' - value should be within the accepted boundaries recommended by the organization

✗ Prevent Service from exposing node port [1 occurrence]
  - metadata.name: jhooq-springboot (kind: Service)
  ⚡ Incorrect value for key 'type' - 'NodePort' will open a port on all nodes where it can be reached by the network external to the cluster

✗ Ensure each container has a configured liveness probe [1 occurrence]
  - metadata.name: jhooq-springboot (kind: Deployment)
  ⚡ Missing property object 'livenessProbe' - add a properly configured livenessProbe to catch possible deadlocks

✗ Ensure each container has a configured readiness probe [1 occurrence]
  - metadata.name: jhooq-springboot (kind: Deployment)
  ⚡ Missing property object 'readinessProbe' - add a properly configured readinessProbe to notify kubelet your Pods are ready for traffic

(Summary)

- Passing YAML validation: 1/1
- Passing Kubernetes (1.18.0) schema validation: 1/1
- Passing policy check: 0/1

+-----+-----+
| Enabled rules in policy "Default" | 21 |
| Configs tested against policy    | 2  |
| Total rules evaluated             | 21 |
| Total rules failed                | 7  |
| Total rules passed                | 14 |
| See all rules in policy           | https://app.datree.io/login?cliId=nar8cAdWwSouxxRGvPLLBH |
+-----+-----+

```

Click on the link

```

+-----+-----+
| Enabled rules in policy "Default" | 21 |
| Configs tested against policy    | 2  |
| Total rules evaluated             | 21 |
| Total rules failed                | 7  |
| Total rules passed                | 14 |
| See all rules in policy           | https://app.datree.io/login?cliId=nar8cAdWwSouxxRGvPLLBH |
+-----+-----+

```

### 3. How to understand the validation errors thrown by Datree

What I have discovered is that my Kubernetes manifest (YAML) has a lot of problems. it will work fine inside my Kubernetes cluster, it is still far from ready for the production environment.

Let's start with the errors -

#### 3.1 Ensure each container has a configured memory request

✗ Ensure each container has a configured memory request [2 occurrences]  
 — metadata.name: RELEASE-NAME-helloworld (kind: Deployment)

— metadata.name: RELEASE-NAME-helloworld-test-connection (kind: Pod)

💡 Missing property object `requests.memory` - value should be within the accepted boundaries recommended by the organization

This error tells me that I have neglected to add request.memory size inside the Kubernetes deployment manifest.

This error can easily be fixed by adding the following request.memory attribute - resources:

requests:

memory: "128Mi"

cpu: "512m"

### 3.2 Ensure each container has a configured CPU request

✗ Ensure each container has a configured CPU request [2 occurrences]

— metadata.name: RELEASE-NAME-helloworld (kind: Deployment)

— metadata.name: RELEASE-NAME-helloworld-test-connection (kind: Pod)

💡 Missing property object `requests.cpu` - value should be within the accepted boundaries recommended by the organization

Similar to the previous error 3.1, we need to add the limit.cpu attribute.

Here is the missing limit.cpu attribute along with request.memory -

resources:

requests:

memory: "128Mi"

cpu: "512m"

### 3.3 Ensure each container has a configured memory limit

✗ Ensure each container has a configured memory limit [2 occurrences]

— metadata.name: RELEASE-NAME-helloworld-test-connection (kind: Pod)

— metadata.name: RELEASE-NAME-helloworld (kind: Deployment)

💡 Missing property object `limits.memory` - value should be within the accepted boundaries recommended by the organization

After fixing the memory and cpu requests, let's try to fix the memory limit.

Add the following memory snippet inside your deployment manifest of Kubernetes under the resource limits -

limits:

```
memory: "128Mi"
```

### 3.4 Ensure each container has a configured CPU limit

- ✗ Ensure each container has a configured CPU limit [2 occurrences]
  - metadata.name: RELEASE-NAME-helloworld-test-connection (kind: Pod)
  - metadata.name: RELEASE-NAME-helloworld (kind: Deployment)
- 💡 Missing property object `limits.cpu` - value should be within the accepted boundaries recommended by the organization

You also need to update the memory for the cpu, which can be done by adding the following attributes along with the previous one 3.4 -

limits:

```
memory: "128Mi"
```

```
cpu: "512m"
```

### 3.5 Ensure each container has a configured liveness probe

- ✗ Ensure each container has a configured liveness probe [1 occurrence]
  - metadata.name: RELEASE-NAME-helloworld-test-connection (kind: Pod)
- 💡 Missing property object `livenessProbe` - add a properly configured livenessProbe to catch possible deadlocks

*MYSQL*

Next on the list are the liveness and readiness probes. They go hand in hand, but to keep the things simple and easy to understand, we will look at each error individually.

livenessProbe:

```
httpGet:
```

```
  path: /hello
```


```
  port: 8080
```

```
initialDelaySeconds: 15
```

```
periodSeconds: 10
```

### 3.6 Ensure each container has a configured readiness probe

- ✗ Ensure each container has a configured readiness probe [1 occurrence]
  - metadata.name: RELEASE-NAME-helloworld-test-connection (kind: Pod)

 Missing property object `readinessProbe` - **add** a properly configured readiness Probe **to** notify kubelet your Pods are ready **for** traffic

*MYSQL*

After fixing the liveness probe, let's fix the readiness probe by adding the following attribute to the Kubernetes manifest (YAML) -

readinessProbe:

httpGet:


path: /hello


port: 8080

initialDelaySeconds: 15

periodSeconds: 10

### 3.7 Ensure Deployment has more than one replica configured

 Ensure Deployment has more than one replica configured [1 occurrence]  
— metadata.name: **RELEASE-NAME-helloworld** (kind: Deployment)

 Incorrect value **for key** `replicas` - running **2 or** more replicas will increase the availability of the service

*MYSQL*

Lastly, we need to fix the replica count. This can be done easily by updating the replicas attribute.

It is not recommended to have a replica count of 1. Update the replica count to a number greater than 1 -

spec:

replicas: 2

**Here is the final k8s-spring-boot-deployment.yml after all 7 fixes have been made:**

apiVersion: apps/v1

kind: Deployment

metadata:

name: jhooq-springboot

spec:

replicas: 2

selector:

matchLabels:

app: jhooq-springboot

```
template:
  metadata:
    labels:
      app: jhooq-springboot
  spec:
    containers:
      - name: springboot
        image: rahulwagh17/kubernetes:jhooq-k8s-springboot

    resources:
      requests:
        memory: "128Mi"
        cpu: "512m"
      limits:
        memory: "128Mi"
        cpu: "512m"

    ports:
      - containerPort: 8080

    readinessProbe:
      httpGet:
        path: /hello
        port: 8080
      initialDelaySeconds: 15
      periodSeconds: 10

    livenessProbe:
      httpGet:
        path: /hello
        port: 8080
      initialDelaySeconds: 15
      periodSeconds: 10

    env:
      - name: PORT
        value: "8080"
```

---

```
apiVersion: v1
kind: Service
```



```
metadata:
  name: jhooq-springboot
spec:
  type: ClusterIP
  ports:
    - port: 80
      targetPort: 8080
  selector:
    app: jhooq-springboot
...
```

I recommend updating your Kubernetes manifest and re-running the Datree test command.

```
# datree test k8s-spring-boot-deployment.yml
```

Here is the screenshot of the results, where you can see that all the validation errors are now gone.

```
$ datree test k8s-spring-boot-deployment.yml

(Summary)

- Passing YAML validation: 1/1
- Passing Kubernetes (1.18.0) schema validation: 1/1
- Passing policy check: 1/1

+-----+-----+
| Enabled rules in policy "Default" | 21 |
| Configs tested against policy      | 2  |
| Total rules evaluated              | 21 |
| Total rules failed                  | 0  |
| Total rules passed                  | 21 |
| See all rules in policy             | https://app.datree.io/login?cliId=nar8cAdWwSouxxRGvPLLBH |
+-----+-----+
```

Copy the URL from the terminal and paste it in your browser. Authenticate yourself using your GitHub account and then you will see the following dashboard

—

