SpringBootHelm-2

Saturday, January 8, 2022 8:54 PM

deployment.yaml

The next configuration we need to update is deployment.yaml and as the name suggest it is used for deployment purpose. So lets see how does it looks like

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ include "springboot.fullname" . }}
  labels:
    {{- include "springboot.labels" . | nindent 4 }}
spec:
{{- if not .Values.autoscaling.enabled }}
  replicas: {{ .Values.replicaCount }}
{{- end }}
  selector:
    matchLabels:
      {{- include "springboot.selectorLabels" . | nindent 6 }}
  template:
    metadata:
    {{- with .Values.podAnnotations }}
      annotations:
        {{- toYaml . | nindent 8 }}
    {{- end }}
      labels:
        {{- include "springboot.selectorLabels" . | nindent 8 }}
    spec:
      {{- with .Values.imagePullSecrets }}
      imagePullSecrets:
        {{- toYaml . | nindent 8 }}
      {{- end }}
      serviceAccountName: {{ include "springboot.serviceAccountName" . }}
      securityContext:
        {{- toYaml .Values.podSecurityContext | nindent 8 }}
      containers:
        - name: {{ .Chart.Name }}
          securityContext:
            {{- toYaml .Values.securityContext | nindent 12 }}
          image: "{{ .Values.image.repository }}"   #update here
          imagePullPolicy: {{ .Values.image.pullPolicy }}
          ports:
            - name: http
              containerPort: 8080     #update here
              protocol: TCP          #comment it
              #livenessProbe:          #comment it
              #httpGet:          #comment it
```

```
          #path: /          #comment it
          #port: http          #comment it
          #readinessProbe:          #comment it
          #httpGet:          #comment it
          #path: /          #comment it
          #port: http          #comment it
        resources:
          {{- toYaml .Values.resources | nindent 12  }}
    {{- with .Values.nodeSelector }}
    nodeSelector:
      {{- toYaml . | nindent 8 }}
    {{- end }}
    {{- with .Values.affinity }}
    affinity:
      {{- toYaml . | nindent 8 }}
    {{- end }}
    {{- with .Values.tolerations }}
    tolerations:
      {{- toYaml . | nindent 8 }}
    {{- end }}
```

Do not get scared we do not need to update each and every configuration. But we need to update the **containerPort** to 8080

1. **containerPort : 8080**

Because we need to deploy our spring boot application at port 8080.

service.yaml

The service.yaml is basically used for exposing our kubernetes springboot deployment as service. The good thing over here we do not need to update any configuration here.

```
apiVersion: v1
kind: Service
metadata:
  name: {{ include "springboot.fullname" . }}
  labels:
    {{- include "springboot.labels" . | nindent 4 }}
spec:
  type: {{ .Values.service.type }}
  ports:
    - port: {{ .Values.service.port }}
      targetPort: http
      protocol: TCP
      name: http
  selector:
    {{- include "springboot.selectorLabels" . | nindent 4 }}
```

4. Run $ helm template springboot

Now we are into the step no 4 and we are pretty much ready with our first Helm Chart for our spring boot application.

But wait - *Wouldn't it be nice if you could see the service.yaml, deployment.yaml with its actual values before running the helm install command*?

Yes you can do that with the helm template command -

```bash
helm template springboot
```

(*You can not run the above command from inside the springboot directory, so you should get out from the springboot directory and then execute the command*)

After running the above command it should return you will service.yaml, deployment.yaml and test-connection.yaml with actual values

```
---
# Source: springboot/templates/serviceaccount.yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: RELEASE-NAME-springboot
  labels:
    helm.sh/chart: springboot-0.1.0
    app.kubernetes.io/name: springboot
    app.kubernetes.io/instance: RELEASE-NAME
    app.kubernetes.io/version: "1.16.0"
    app.kubernetes.io/managed-by: Helm
---
# Source: springboot/templates/service.yaml
apiVersion: v1
kind: Service
metadata:
  name: RELEASE-NAME-springboot
  labels:
    helm.sh/chart: springboot-0.1.0
    app.kubernetes.io/name: springboot
    app.kubernetes.io/instance: RELEASE-NAME
    app.kubernetes.io/version: "1.16.0"
    app.kubernetes.io/managed-by: Helm
spec:
```

```
  type: ClusterIP
  ports:
    - port: 8080
      targetPort: http
      protocol: TCP
      name: http
  selector:
    app.kubernetes.io/name: springboot
    app.kubernetes.io/instance: RELEASE-NAME
---
# Source: springboot/templates/deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: RELEASE-NAME-springboot
  labels:
    helm.sh/chart: springboot-0.1.0
    app.kubernetes.io/name: springboot
    app.kubernetes.io/instance: RELEASE-NAME
    app.kubernetes.io/version: "1.16.0"
    app.kubernetes.io/managed-by: Helm
spec:
  replicas: 1
  selector:
    matchLabels:
      app.kubernetes.io/name: springboot
      app.kubernetes.io/instance: RELEASE-NAME
  template:
    metadata:
      labels:
        app.kubernetes.io/name: springboot
        app.kubernetes.io/instance: RELEASE-NAME
    spec:
      serviceAccountName: RELEASE-NAME-springboot
      securityContext:
        {}
      containers:
        - name: springboot
          securityContext:
            {}
          image: "rahulwagh17/kubernetes:jhooq-k8s-springboot"
          imagePullPolicy: IfNotPresent
          ports:
            - name: http
              containerPort: 8080
              protocol: TCP
          resources:
            {}
---
# Source: springboot/templates/tests/test-connection.yaml
apiVersion: v1
kind: Pod
metadata:
```

```
  name: "RELEASE-NAME-springboot-test-connection"
  labels:
    helm.sh/chart: springboot-0.1.0
    app.kubernetes.io/name: springboot
    app.kubernetes.io/instance: RELEASE-NAME
    app.kubernetes.io/version: "1.16.0"
    app.kubernetes.io/managed-by: Helm
  annotations:
    "helm.sh/hook": test-success
spec:
  containers:
    - name: wget
      image: busybox
      command: ['wget']
      args: ['RELEASE-NAME-springboot:8080']
  restartPolicy: Never
```

I love this command because it locally renders the templates by replacing all the placeholders with its actual values.

There is one more sanitary command **lint** provided by helm which you could run to identify possible issues forehand.

```bash
helm lint springboot
```

```bash
==> Linting springboot
[INFO] Chart.yaml: icon is recommended

1 chart(s) linted, 0 chart(s) failed
```

5. helm -debug -dry-run

The next check which we are going to do is **-dry-run**. Helm is full of such useful utility which allows developer to test its configuration before running the final install command

Use the following **-dry-run** command to verify your Spring Boot Helm Chart

```bash
helm install springboot --debug --dry-run springboot
```

If there is something wrong with your Helm chart configuration then it will going to prompt you immediately.

6. helm install

Alright lets run the install command.

```bash
helm install myfirstspringboot springboot
```

```
vagrant@node1:~$ helm install myfirstspringboot springboot
NAME: myfirstspringboot
LAST DEPLOYED: Tue Jul  7 17:25:59 2020
NAMESPACE: default
STATUS: deployed
REVISION: 1
```

Lets break down the command because if you are doing it for the first time then it might be little confusing for you.

1. **myfirstspringboot** : It's a release name for helm chart otherwise helm will generate its own release name that is why we have assigned this name.
2. **springboot** : It is our actual chart name which we created in Step 2.

7. Verify the helm install

Next question which comes into my mind - "**How to see all the releases? **"

Use the following list command to list down all the releases -

```bash
helm list -a
```

```
vagrant@node1:~$ helm list -a
NAME                    NAMESPACE       REVISION        UPDATED                                 STATUS          CHART                   APP VERSION
myfirstspringboot       default         1               2020-07-07 17:25:59.653915661 +0000 UTC deployed        springboot-0.1.0        1.16.0
```

Lets do some cross verification using kubectl commands also, so that we can make sure helm has done its work.

```bash
kubectl get all
```

```
vagrant@node1:~/springboot$ kubectl get all
NAME                                 READY   STATUS    RESTARTS   AGE
pod/myfirstspringboot-5987d9c7df-r6gk5   1/1     Running   0          3m41s

NAME                         TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
service/kubernetes           ClusterIP   10.233.0.1      <none>        443/TCP    28d
service/myfirstspringboot    ClusterIP   10.233.28.240   <none>        8080/TCP   3m41s

NAME                                 READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/myfirstspringboot    1/1     1            1           3m41s

NAME                                           DESIRED   CURRENT   READY   AGE
replicaset.apps/myfirstspringboot-5987d9c7df   1         1         1       3m41s
```

```bash
helm upgrade myfirstspringboot .
```

```
vagrant@node1:~/springboot$ helm upgrade myfirstspringboot .
Release "myfirstspringboot" has been upgraded. Happy Helming!
NAME: myfirstspringboot
LAST DEPLOYED: Tue Jul  7 19:11:23 2020
NAMESPACE: default
STATUS: deployed
REVISION: 2
NOTES:
1. Get the application URL by running these commands:
  export POD_NAME=$(kubectl get pods --namespace default -l "app.kubernete
s.io/name=springboot,app.kubernetes.io/instance=myfirstspringboot" -o json
path="{.items[0].metadata.name}")
  echo "Visit http://127.0.0.1:8080 to use your application"
  kubectl --namespace default port-forward $POD_NAME 8080:80
```

You should see a message with your release name .i.e. - ***Release "myfirstspringboot" has been upgraded. Happy Helming!***

Verify your helm upgrade by running following list command

```
vagrant@node1:~/springboot$ helm list -a
NAME                    NAMESPACE       REVISION
myfirstspringboot       default         2
```

As you can see now the revision count is 2.

Lets again run kubectl get deployment

```
kubectl get deployments
```

```
vagrant@node1:~/springboot$ kubectl get deployments
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
myfirstspringboot   2/2     2            2           33m
```

And now you can see we have successfully upgraded the replica count from 1 to 2.

10. Delete Helm release

Wouldn't it be nice if you need to run only one command to delete your Helm release and you do not have to do anything else.

BASH

```
helm delete myfirstspringboot
```

As you can see now you have successfully delete your release with single **helm delete** command.

```
vagrant@node1:~/springboot$ helm delete myfirstspringboot
release "myfirstspringboot" uninstalled
```