

Deploying Apache HTTPD on Kubernetes Using Helm (With Lightweight Kubernetes Setup)

Introduction to Helm

Helm is a package manager for Kubernetes that simplifies the deployment and management of applications. Instead of writing multiple YAML files and applying them manually using `kubectl apply -f`, Helm allows you to package all Kubernetes resources into a single **Helm Chart**.

Using Helm, we can define variables in `values.yaml` and reuse them dynamically instead of hardcoding values in multiple YAML files.

Prerequisites

Before proceeding, ensure you have the following:

- An AWS EC2 instance (Ubuntu 20.04 or later)

- K3s (Lightweight Kubernetes) installed

- Helm installed

Step-by-Step Deployment Guide

Step 1: Install a Lightweight Kubernetes (K3s) on Your EC2 Instance

Since running a full Kubernetes cluster (e.g., using `kubeadm`) can be resource-intensive, we'll use **K3s**, a lightweight Kubernetes distribution that is easy to install and well-suited for EC2 instances.

1. Install K3s

Run the following command on your EC2 instance:

```
curl -sfL https://get.k3s.io | sh -
```

2. Verify Kubernetes Installation

After installation, check if Kubernetes is running:

```
kubectl get nodes
```

```

ubuntu@ip-172-31-1-249:~$
ubuntu@ip-172-31-1-249:~$
ubuntu@ip-172-31-1-249:~$ curl -sL https://get.k3s.io | sh -
[INFO] Finding release for channel stable
[INFO] Using v1.31.6+k3s1 as release
[INFO] Downloading hash https://github.com/k3s-io/k3s/releases/download/v1.31.6+k3s1/sha256sum-amd64.txt
[INFO] Downloading binary https://github.com/k3s-io/k3s/releases/download/v1.31.6+k3s1/k3s
[INFO] Verifying binary download
[INFO] Installing k3s to /usr/local/bin/k3s
[INFO] Skipping installation of SELinux REM
[INFO] Creating /usr/local/bin/kubectl symlink to k3s
[INFO] Creating /usr/local/bin/crictl symlink to k3s
[INFO] Creating /usr/local/bin/ctr symlink to k3s
[INFO] Creating killall script /usr/local/bin/k3s-killall.sh
[INFO] Creating uninstall script /usr/local/bin/k3s-uninstall.sh
[INFO] env: Creating environment file /etc/systemd/system/k3s.service.env
[INFO] systemd: Creating service file /etc/systemd/system/k3s.service
[INFO] systemd: Enabling k3s unit
Created symlink /etc/systemd/system/multi-user.target.wants/k3s.service → /etc/systemd/system/k3s.service.
[INFO] systemd: Starting k3s
ubuntu@ip-172-31-1-249:~$ sudo kubectl get nodes
NAME                STATUS    ROLES    AGE   VERSION
ip-172-31-1-249    Ready    control-plane,master   4s    v1.31.6+k3s1
ubuntu@ip-172-31-1-249:~$

```

If kubectl is not found, set up your environment:

```
export KUBECONFIG=/etc/rancher/k3s/k3s.yaml
```

Now check again:

```
kubectl get nodes
```

You should see your EC2 instance listed as a Kubernetes node.

Expected Output:

```
...
```

NAME	STATUS	ROLES	AGE	VERSION
ip-172-31-1-249	Ready	control-plane,master	5m	v1.31.6+k3s1

```
...
```

Step 2: Install Helm on Your EC2 Instance

If Helm is not installed, run the following commands:

```
curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
```

```
chmod 700 get_helm.sh
```

```
./get_helm.sh
```

Verify the installation:

helm version

Session ID: sandeep-8zeoxlqc8dxxegkfrvgjqc5a8

Instance ID: i-07135cf55ef5949a8

```
buntu@ip-172-31-1-249:~$ curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
buntu@ip-172-31-1-249:~$ chmod 700 get_helm.sh
buntu@ip-172-31-1-249:~$ ./get_helm.sh
Downloading https://get.helm.sh/helm-v3.17.2-linux-amd64.tar.gz
Verifying checksum... Done.
Preparing to install helm into /usr/local/bin
helm installed into /usr/local/bin/helm
buntu@ip-172-31-1-249:~$ helm version
version.BuildInfo{Version:"v3.17.2", GitCommit:"ec0bbbd6d6276b83880042c1ecb34087e84d41eb", GitTreeState:"clean", GoVersion:"go1.23.7"}
buntu@ip-172-31-1-249:~$
```

Step 3: Create a Helm Chart for HTTPD

helm create httpd-chart

cd httpd-chart

This command generates a folder structure under httpd-chart/. Now, modify the necessary files.

Session ID: sandeep-8zeoxlqc8dxxegkfrvgjqc5a8

Instance ID: i-07135cf55ef5949a8

```
buntu@ip-172-31-1-249:~$ helm create httpd-chart
Creating httpd-chart
buntu@ip-172-31-1-249:~$ cd httpd-chart/
buntu@ip-172-31-1-249:~/httpd-chart$ ls -altr
total 32
-rw-r--r-- 1 ubuntu ubuntu 4297 Mar 20 13:18 values.yaml
drwxr-xr-x 3 ubuntu ubuntu 4096 Mar 20 13:18 templates
drwxr-xr-x 2 ubuntu ubuntu 4096 Mar 20 13:18 charts
-rw-r--r-- 1 ubuntu ubuntu 1147 Mar 20 13:18 Chart.yaml
-rw-r--r-- 1 ubuntu ubuntu 349 Mar 20 13:18 .helmignore
drwxr-xr-x 4 ubuntu ubuntu 4096 Mar 20 13:18 ..
drwxr-xr-x 4 ubuntu ubuntu 4096 Mar 20 13:18 .
buntu@ip-172-31-1-249:~/httpd-chart$
```

Step 4: Modify Helm Chart Files

```

ubuntu@ip-172-31-1-249: /var/snap/amazon-ssm-agent/9881$ cd
ubuntu@ip-172-31-1-249:~$
ubuntu@ip-172-31-1-249:~$
ubuntu@ip-172-31-1-249:~$ cd httpd-chart/
ubuntu@ip-172-31-1-249:~/httpd-chart$ ll
total 32
-rwxr-xr-x 4 ubuntu ubuntu 4096 Mar 20 13:18 ./
-rwxr-xr-x 4 ubuntu ubuntu 4096 Mar 20 13:19 ../
-rw-r--r-- 1 ubuntu ubuntu 349 Mar 20 13:18 .helmignore
-rw-r--r-- 1 ubuntu ubuntu 1147 Mar 20 13:18 Chart.yaml
-rwxr-xr-x 2 ubuntu ubuntu 4096 Mar 20 13:18 charts/
-rwxr-xr-x 3 ubuntu ubuntu 4096 Mar 20 13:18 templates/
-rw-r--r-- 1 ubuntu ubuntu 4297 Mar 20 13:18 values.yaml
ubuntu@ip-172-31-1-249:~/httpd-chart$ vi values.yaml
ubuntu@ip-172-31-1-249:~/httpd-chart$ cp -prf values.yaml values.yaml.orig
ubuntu@ip-172-31-1-249:~/httpd-chart$ vi values.yaml
ubuntu@ip-172-31-1-249:~/httpd-chart$ cd templates/
ubuntu@ip-172-31-1-249:~/httpd-chart/templates$ ll
total 40
-rwxr-xr-x 3 ubuntu ubuntu 4096 Mar 20 13:18 ./
-rwxr-xr-x 4 ubuntu ubuntu 4096 Mar 20 13:19 ../
-rw-r--r-- 1 ubuntu ubuntu 1760 Mar 20 13:18 NOTES.txt
-rw-r--r-- 1 ubuntu ubuntu 1822 Mar 20 13:18 helpers.tpl
-rw-r--r-- 1 ubuntu ubuntu 2400 Mar 20 13:18 deployment.yaml
-rw-r--r-- 1 ubuntu ubuntu 1003 Mar 20 13:18 hpa.yaml
-rw-r--r-- 1 ubuntu ubuntu 1100 Mar 20 13:18 ingress.yaml
-rw-r--r-- 1 ubuntu ubuntu 373 Mar 20 13:18 service.yaml
-rw-r--r-- 1 ubuntu ubuntu 397 Mar 20 13:18 serviceaccount.yaml
-rwxr-xr-x 2 ubuntu ubuntu 4096 Mar 20 13:18 tests/
ubuntu@ip-172-31-1-249:~/httpd-chart/templates$ vi deployment.yaml
ubuntu@ip-172-31-1-249:~/httpd-chart/templates$ vi service
ubuntu@ip-172-31-1-249:~/httpd-chart/templates$ vi service.yaml
ubuntu@ip-172-31-1-249:~/httpd-chart/templates$ vi ingress.yaml
ubuntu@ip-172-31-1-249:~/httpd-chart/templates$

```

1. values.yaml (Configuration for Deployment, Service, and Ingress)

Modify `values.yaml` to configure the deployment settings

2. templates/deployment.yaml (Deployment for HTTPD)

3. templates/service.yaml (Service to Expose HTTPD)

4. templates/ingress.yaml (Ingress for External Access - Optional)

Step 5: Deploy the Helm Chart

```
sudo cp -prf /etc/rancher/k3s/k3s.yaml /home/ubuntu/kube.conf
```

```
sudo chown ubuntu:ubuntu /home/ubuntu/kube.conf
```

```
export KUBECONFIG=/home/ubuntu/kube.conf
```

```
kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
------	--------	-------	-----	---------

ip-172-31-1-249	Ready	control-plane,master	148m	v1.31.6+k3s1
-----------------	-------	----------------------	------	--------------

```

ubuntu@ip-172-31-1-249:~$ sudo cp -prf /etc/rancher/k3s/k3s.yaml /home/ubuntu/kube.conf
ubuntu@ip-172-31-1-249:~$ sudo chown ubuntu:ubuntu /home/ubuntu/kube.conf
ubuntu@ip-172-31-1-249:~$ cd /home/ubuntu/
ubuntu@ip-172-31-1-249:~$ ls -altr
total 60
-rw-r--r-- 1 ubuntu ubuntu 807 Mar 31 2024 .profile
-rw-r--r-- 1 ubuntu ubuntu 3771 Mar 31 2024 .bashrc
-rw-r--r-- 1 ubuntu ubuntu 220 Mar 31 2024 .bash_logout
drwxr----- 2 ubuntu ubuntu 4096 Mar 20 13:08 .ssh
drwxr-xr-x 4 root root 4096 Mar 20 13:08 ..
-rw-r--r-- 1 ubuntu ubuntu 0 Mar 20 13:09 .sudo_as_admin_successful
-rw----- 1 ubuntu ubuntu 2957 Mar 20 13:12 kube.conf
-rw----- 1 ubuntu ubuntu 11913 Mar 20 13:16 get_helm.sh
-rw----- 1 ubuntu ubuntu 329 Mar 20 13:39 .bash_history
drwxr-xr-x 4 ubuntu ubuntu 4096 Mar 20 15:33 httpd-chart
-rw----- 1 ubuntu ubuntu 9436 Mar 20 15:35 .viminfo
drwxr-xr-x 4 ubuntu ubuntu 4096 Mar 20 15:39 .
ubuntu@ip-172-31-1-249:~$ export KUBECONFIG=/home/ubuntu/kube.conf
ubuntu@ip-172-31-1-249:~$ kubectl get nodes
NAME                STATUS    ROLES    AGE     VERSION
ip-172-31-1-249     Ready    control-plane,master   148m   v1.31.6+k3s1
ubuntu@ip-172-31-1-249:~$

```

1. Install the chart
2. helm install my-httpd ./httpd-chart

```

ubuntu@ip-172-31-1-249:~/httpd-chart/templates$
ubuntu@ip-172-31-1-249:~/httpd-chart/templates$ cd
ubuntu@ip-172-31-1-249:~$ ls -altr httpd-chart/templates/
total 32
drwxr-xr-x 2 ubuntu ubuntu 4096 Mar 20 13:18 tests
-rw-r--r-- 1 ubuntu ubuntu 1622 Mar 20 13:18 _helpers.tpl
-rw-r--r-- 1 ubuntu ubuntu 1769 Mar 20 13:18 NOTES.txt
drwxr-xr-x 4 ubuntu ubuntu 4096 Mar 20 15:33 ..
-rw-r--r-- 1 ubuntu ubuntu 396 Mar 20 15:34 deployment.yaml
-rw-r--r-- 1 ubuntu ubuntu 219 Mar 20 15:34 service.yaml
-rw-r--r-- 1 ubuntu ubuntu 414 Mar 20 15:35 ingress.yaml
drwxr-xr-x 3 ubuntu ubuntu 4096 Mar 20 15:45 .
ubuntu@ip-172-31-1-249:~$ helm install my-httpd ./httpd-chart
NAME: my-httpd
LAST DEPLOYED: Thu Mar 20 15:46:08 2025
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
1. Get the application URL by running these commands:
export POD_NAME=$(kubectl get pods --namespace default -l "app.kubernetes.io/name=httpd-chart,app.kubernetes.io/instance=my-httpd" -o jsonpath="{.items[0].metadata.name}")
export CONTAINER_PORT=$(kubectl get pod --namespace default $POD_NAME -o jsonpath="{.spec.containers[0].ports[0].containerPort}")
echo "Visit http://127.0.0.1:8080 to use your application"
kubectl --namespace default port-forward $POD_NAME 8080:$CONTAINER_PORT
ubuntu@ip-172-31-1-249:~$

```

3. Check if the deployment is running
4. kubectl get pods
5. kubectl get svc

```

ubuntu@ip-172-31-1-249:~$ kubectl get pods
NAME                READY    STATUS    RESTARTS    AGE
httpd-deployment-bf5c4df59-mhcb6   1/1      Running   0            83s
ubuntu@ip-172-31-1-249:~$ kubectl get svc
NAME                TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
httpd-service       ClusterIP     10.43.151.81   <none>          80/TCP     116s
kubernetes           ClusterIP     10.43.0.1      <none>          443/TCP    155m
ubuntu@ip-172-31-1-249:~$

```

6. Test the service
7. kubectl port-forward svc/httpd-service 8080:80

```
Session ID: sandeep-xaiye2qet82ibaxia3ppq4f8 Instance ID: i-07135cf55ef5949a8

ubuntu@ip-172-31-1-249:~$ kubectl port-forward svc/httpd-service 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::]:8080 -> 80
Handling connection for 8080
```

8. curl <http://localhost:8080>

```
Session ID: sandeep-8lbp62fq4iqbv32f6sqaabz9y Instance ID: i-07135cf55ef5949a8

$ sudo su ubuntu
ubuntu@ip-172-31-1-249:~$ cd /var/snap/amazon-ssm-agent/9881$ curl http://localhost:8080
<html><body><h1>It works!</h1></body></html>
ubuntu@ip-172-31-1-249:~$
```

Step 6: Enable Ingress (Optional: Expose HTTPD Externally)

If you want to expose the service using Ingress, follow these steps:

1. Install Nginx Ingress Controller

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/main/deploy/static/provider/cloud/deploy.yaml
```

2. Edit `values.yaml` and set `ingress.enabled` to true

3. Upgrade the Helm deployment:

```
helm upgrade my-httpd ./httpd-chart
```

```

ubuntu@ip-172-31-1-249:~$ kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/main/deploy/static/provider/cloud/deploy.yaml
namespace/ingress-nginx created
serviceaccount/ingress-nginx created
serviceaccount/ingress-nginx-admission created
role.rbac.authorization.k8s.io/ingress-nginx created
role.rbac.authorization.k8s.io/ingress-nginx-admission created
clusterrole.rbac.authorization.k8s.io/ingress-nginx created
clusterrole.rbac.authorization.k8s.io/ingress-nginx-admission created
rolebinding.rbac.authorization.k8s.io/ingress-nginx created
rolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
configmap/ingress-nginx-controller created
service/ingress-nginx-controller created
service/ingress-nginx-controller-admission created
deployment.apps/ingress-nginx-controller created
job.batch/ingress-nginx-admission-create created
job.batch/ingress-nginx-admission-patch created
ingressclass.networking.k8s.io/nginx created
validatingwebhookconfiguration.admissionregistration.k8s.io/ingress-nginx-admission created
ubuntu@ip-172-31-1-249:~$ grep -i ingress.enabled httpd-chart/values.yaml
ubuntu@ip-172-31-1-249:~$ grep -i ingress httpd-chart/values.yaml
ingress:
  enabled: false # Change to true if using Ingress
ubuntu@ip-172-31-1-249:~$ vi httpd-chart/values.yaml
ubuntu@ip-172-31-1-249:~$ grep -i ingress httpd-chart/values.yaml
ingress:
  enabled: true # Change to true if using Ingress
ubuntu@ip-172-31-1-249:~$ he
head      helm      help      helptags  hexdump
ubuntu@ip-172-31-1-249:~$ helm upgrade my-httpd ./httpd-chart
Release "my-httpd" has been upgraded. Happy Helming!
NAME: my-httpd
LAST DEPLOYED: Thu Mar 20 15:55:05 2025
NAMESPACE: default
STATUS: deployed
REVISION: 2
NOTES:
1. Get the application URL by running these commands:
ubuntu@ip-172-31-1-249:~$

```

4. Update /etc/hosts on your EC2 instance:

5. echo "\$(kubectl get nodes -o jsonpath='{.items[0].status.addresses[0].address}')" httpd.local" | sudo tee -a /etc/hosts

Session ID: sandeep-xaiye2qk4t82lbaiclx9rrpqk48

Instance ID: i-07135cf5ef5949a8

```

ubuntu@ip-172-31-1-249:~$ cat /etc/hosts
127.0.0.1 localhost

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
ubuntu@ip-172-31-1-249:~$ echo "$(kubectl get nodes -o jsonpath='{.items[0].status.addresses[0].address}')" httpd.local" | sudo tee -a /etc/hosts
172.31.1.249 httpd.local
ubuntu@ip-172-31-1-249:~$ cat /etc/hosts
127.0.0.1 localhost

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
172.31.1.249 httpd.local
ubuntu@ip-172-31-1-249:~$

```

6. Test the HTTPD service using the hostname

7. curl <http://httpd.local>

```

ubuntu@ip-172-31-1-249:~$ kubectl get svc -n ingress-nginx
NAME                                TYPE                CLUSTER-IP      EXTERNAL-IP      PORT(S)                                AGE
ingress-nginx-controller            LoadBalancer        10.43.106.242    <pending>         80:31788/TCP,443:31804/TCP            91m
ingress-nginx-controller-admission  ClusterIP            10.43.96.152     <none>            443/TCP                                91m
ubuntu@ip-172-31-1-249:~$
ubuntu@ip-172-31-1-249:~$
ubuntu@ip-172-31-1-249:~$ curl http://httpd.local
<html><body><h1>It works!</h1></body></html>
ubuntu@ip-172-31-1-249:~$

```

Step 7: Cleanup

To uninstall the Helm release:

```
helm uninstall my-httpd
```

To remove K3s (if needed):

```
/usr/local/bin/k3s-uninstall.sh
```

Conclusion

Using **K3s** (lightweight Kubernetes), we deployed an **Apache HTTPD service** on Kubernetes using Helm. Unlike kubectl apply -f, Helm makes deployments **reusable, configurable, and easy to maintain**.