Deploying a production-grade WordPress application on Kubernetes using Docker and Helm charts

Introduction

This documentation provides a comprehensive guide for deploying a production-grade WordPress application on Kubernetes using Docker and Helm charts. The project includes custom Docker images for WordPress, MySQL, and Nginx, along with Kubernetes resources for PersistentVolumes (PVs), PersistentVolumeClaims (PVCs), deployments, and services.

Table of Contents

Prerequisites

- Minikube installed and running
- Docker installed and configured
- Helm installed and configured
- Access to a Docker repository (Docker Hub, Google Container Registry, etc.)

Project Structure

The project structure includes the following directories and files:

```
|       └── hello.lua
├── Wordpress
|   └── Dockerfile
|   └── custom-php.ini
├── wordpress-chart
|   ├── Chart.yaml
|   ├── values.yaml
|   └── templates
|       ├── deployment.yaml
|       ├── service.yaml
|       └── ingress.yaml
├── pvs.yaml
├── pvcs.yaml
└── README.md
```

Step 1: Set Up PersistentVolumes and PersistentVolumeClaims

We will create PVs and PVCs with ReadWriteMany access mode.

pvs.yaml

This file defines the PersistentVolumes for WordPress and MySQL.

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: wordpress-pv
spec:
  capacity:
    storage: 20Gi
  accessModes:
    - ReadWriteMany
  nfs:
    path: /path/to/nfs
    server: nfs-server.example.com
```

```yaml
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mysql-pv
spec:
  capacity:
    storage: 20Gi
  accessModes:
    - ReadWriteMany
  nfs:
    path: /path/to/nfs
    server: nfs-server.example.com
```

Explanation:

- **apiVersion**: Specifies the version of the Kubernetes API.
- **kind**: Defines the type of Kubernetes object, in this case, a PersistentVolume.
- **metadata**: Metadata for the PersistentVolume, including its name.
- **spec**: Specification for the PersistentVolume, including capacity, access modes, and NFS configuration.

pvcs.yaml

This file defines the PersistentVolumeClaims for WordPress and MySQL.

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: wordpress-pvc
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 20Gi
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mysql-pvc
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 20Gi
```

```
```

Explanation:

- **apiVersion**: Specifies the version of the Kubernetes API.
- **kind**: Defines the type of Kubernetes object, in this case, a PersistentVolumeClaim.
- **metadata**: Metadata for the PersistentVolumeClaim, including its name.
- **spec**: Specification for the PersistentVolumeClaim, including access modes and resource requests.

Apply the PV and PVC configurations:

kubectl apply -f pvs.yaml

kubectl apply -f pvcs.yaml

Step 2: Build Docker Images

Build the required Docker images for WordPress, MySQL, and Nginx.

MySQL

mysql/Dockerfile

```
FROM mysql:8.0

COPY my.cnf /etc/mysql/my.cnf

ENV MYSQL_ROOT_PASSWORD=root_password
ENV MYSQL_DATABASE=wordpress
ENV MYSQL_USER=wordpress
ENV MYSQL_PASSWORD=wordpress_password
```

Explanation:

- **FROM**: Specifies the base image, in this case, MySQL 8.0.
- **COPY**: Copies the custom MySQL configuration file into the container.
- **ENV**: Sets environment variables for MySQL, including root password, database name, user, and password.

mysql/my.cnf

[mysqld]

sql_mode=NO_ENGINE_SUBSTITUTION

Explanation:

- **[mysqld]**: Section for MySQL daemon configuration.
- **sql_mode**: Sets SQL mode to NO_ENGINE_SUBSTITUTION.

Build the MySQL image:

cd mysql

docker build -t snehgupta/mysql:latest . # yourrepo/mysql:latest

Nginx

nginx/Dockerfile

```
# Start with the latest Alpine image
FROM alpine:latest

# Set OpenResty version
ENV OPENRESTY_VERSION=1.19.9.1

# Install necessary packages and dependencies
RUN apk add --no-cache \
    wget \
    tar \
    gcc \
    libc-dev \
    make \
    openssl-dev \
    pcre-dev \
    zlib-dev \
    perl \
    postgresql-dev

# Download and extract OpenResty
RUN wget https://openresty.org/download/openresty-${OPENRESTY_VERSION}.tar.gz && \
    tar -xzvf openresty-${OPENRESTY_VERSION}.tar.gz && \
    rm openresty-${OPENRESTY_VERSION}.tar.gz && \
    cd openresty-${OPENRESTY_VERSION} && \
    ./configure --prefix=/opt/openresty \
    --with-pcre-jit \
    --with-ipv6 \
    --without-http_redis2_module \
    --with-http_iconv_module \
    --with-http_postgres_module \
    -j8 && \
    make -j8 && \
    make install

# Copy custom Nginx configuration and Lua scripts
COPY nginx.conf /opt/openresty/nginx/conf/nginx.conf
```

```
COPY lua-scripts/ /opt/openresty/nginx/lua-scripts/

# Set the command to run OpenResty
CMD ["/opt/openresty/bin/openresty", "-g", "daemon off;"]
```

Explanation:

- **FROM**: Specifies the base image, in this case, Alpine Linux.
- **ENV**: Sets environment variables, including the OpenResty version.
- **RUN**: Installs necessary packages, downloads and extracts OpenResty, and configures and installs OpenResty.
- **COPY**: Copies custom Nginx configuration and Lua scripts into the container.
- **CMD**: Specifies the command to run OpenResty.

nginx/nginx.conf

```
worker_processes 1;

events {
    worker_connections 1024;
}

http {
    include       mime.types;
    default_type  application/octet-stream;

    sendfile        on;
    keepalive_timeout  65;

    # Load Lua scripts
    lua_package_path "/opt/openresty/nginx/lua-scripts/?.lua;;";

    server {
        listen 80;

        location / {
            proxy_pass http://wordpress;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
        }

        location /lua {
            content_by_lua_file /opt/openresty/nginx/lua-scripts/my_script.lua;
        }
    }
}
```

Explanation:

- **worker_processes**: Specifies the number of worker processes.
- **events**: Configures worker connections.
- **http**: Configures HTTP settings, including mime types, sendfile, and keepalive timeout.
- **server**: Configures the server block, including proxy settings and Lua script loading.

nginx/lua-scripts/hello.lua

```
ngx.say("Hello, Lua!")
```

Explanation:

- **ngx.say**: Outputs "Hello, Lua!" when the /lua endpoint is accessed.

Build the Nginx image:

cd ../nginx

docker build -t snehgupta/nginx:latest . # your-repo instead of snehgupta

WordPress

wordpress/Dockerfile

```
FROM wordpress:latest

# Copy custom PHP configuration
COPY custom-php.ini /usr/local/etc/php/conf.d/
```

Explanation:

- **FROM**: Specifies the base image, in this case, the official WordPress image.
- **COPY**: Copies custom PHP configuration into the container.

wordpress/custom-php.ini

```
upload_max_filesize = 64M
post_max_size = 64M
```

Explanation:

- **upload_max_filesize**: Increases the maximum upload file size to 64MB.
- **post_max_size**: Increases the maximum POST size to 64MB.

Build the WordPress image:

cd ../wordpress

docker build -t snehgupta/wordpress:latest .

Step 3: Push Docker Images to a Repository

Push the Docker images to your repository.

 # Login to your Docker repository

docker login

 # Push MySQL image

docker push snehgupta/mysql:latest

 # Push Nginx image

docker push snehgupta/nginx:latest

 # Push WordPress image

docker push snehgupta/wordpress:latest

Step 4: Deploy WordPress with Helm

Deploy the WordPress application using Helm:

helm install my-release ./wordpress-chart

Create a Helm chart for the WordPress application.

wordpress-chart/Chart.yaml

```
apiVersion: v2
name: wordpress-chart
description: A Helm chart for WordPress with MySQL and Nginx
```

```
version: 0.1.0
appVersion: "1.0"
```

Explanation:

- **apiVersion**: Specifies the version of the Helm chart API.
- **name**: Specifies the name of the chart.
- **description**: Provides a description of the chart.
- **type**: Specifies the type of the chart, in this case, an application.
- **version**: Specifies the version of the chart.
- **appVersion**: Specifies the version of the application.

wordpress-chart/values.yaml

```
# Default values for wordpress-chart.
# This is a YAML-formatted file.
# Declare variables to be passed into your templates.

replicaCount: 1

wordpress:
  image:
    repository: wordpress
    tag: latest
    pullPolicy: IfNotPresent

mysql:
  image:
    repository: mysql
    tag: 8.0
    pullPolicy: IfNotPresent

nginx:
  image:
    repository: nginx
    tag: latest
    pullPolicy: IfNotPresent

imagePullSecrets: []

nameOverride: ""
fullnameOverride: ""

serviceAccount:
  create: true
  automount: true
  annotations: {}
```

```yaml
  name: ""

podAnnotations: {}
podLabels: {}

podSecurityContext: {}
# fsGroup: 2000

securityContext: {}
# capabilities:
#   drop:
#   - ALL
# readOnlyRootFilesystem: true
# runAsNonRoot: true
# runAsUser: 1000

service:
  type: LoadBalancer  # Change to LoadBalancer for external access
  port: 80

ingress:
  enabled: false
  className: ""
  annotations: {}
  hosts: []
  tls: []

resources: {}
# limits:
#   cpu: 100m
#   memory: 128Mi
# requests:
#   cpu: 100m
#   memory: 128Mi

livenessProbe:
  httpGet:
    path: /
    port: http
  initialDelaySeconds: 30
  periodSeconds: 10

readinessProbe:
  httpGet:
    path: /
    port: http
  initialDelaySeconds: 5
  periodSeconds: 10
```

```
autoscaling:
  enabled: false
  minReplicas: 1
  maxReplicas: 100
  targetCPUUtilizationPercentage: 80

# Additional volumes on the output Deployment definition.
volumes: []

# Additional volumeMounts on the output Deployment definition.
volumeMounts: []

nodeSelector: {}

tolerations: []

affinity: {}
```

Explanation:

- **wordpress**: Specifies the WordPress configuration, including image, replicas, service, ingress, and resources.
- **mysql**: Specifies the MySQL configuration, including image, service, and resources.
- **nginx**: Specifies the Nginx configuration, including image, replicas, service, ingress, and resources.
- **persistence**: Specifies the persistence configuration, including enabled status, access mode, size, and storage class.

wordpress-chart/templates/deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ include "wordpress-chart.fullname" . }}
  labels:
    {{- include "wordpress-chart.labels" . | nindent 4 }}
spec:
  {{- if not .Values.autoscaling.enabled }}
  replicas: {{ .Values.replicaCount }}
  {{- end }}
  selector:
    matchLabels:
      {{- include "wordpress-chart.selectorLabels" . | nindent 6 }}
  template:
    metadata:
      {{- with .Values.podAnnotations }}
```

```
    annotations:
      {{- toYaml . | nindent 8 }}
    {{- end }}
    labels:
      {{- include "wordpress-chart.labels" . | nindent 8 }}
      {{- with .Values.podLabels }}
      {{- toYaml . | nindent 8 }}
      {{- end }}
  spec:
    {{- with .Values.imagePullSecrets }}
    imagePullSecrets:
      {{- toYaml . | nindent 8 }}
    {{- end }}
    serviceAccountName: {{ include "wordpress-chart.serviceAccountName" . }}
    securityContext:
      {{- toYaml .Values.podSecurityContext | nindent 8 }}
    containers:
      - name: wordpress
        securityContext:
          {{- toYaml .Values.securityContext | nindent 12 }}
        image: "{{ .Values.wordpress.image.repository }}:{{ .Values.wordpress.image.tag }}"
        imagePullPolicy: {{ .Values.wordpress.image.pullPolicy }}
        ports:
          - name: http
            containerPort: 80
            protocol: TCP
        livenessProbe:
          {{- toYaml .Values.livenessProbe | nindent 12 }}
        readinessProbe:
          {{- toYaml .Values.readinessProbe | nindent 12 }}
        resources:
          {{- toYaml .Values.resources | nindent 12 }}
        volumeMounts:
          {{- toYaml .Values.volumeMounts | nindent 12 }}
    volumes:
      {{- toYaml .Values.volumes | nindent 8 }}
    {{- with .Values.nodeSelector }}
    nodeSelector:
      {{- toYaml . | nindent 8 }}
    {{- end }}
    {{- with .Values.affinity }}
    affinity:
      {{- toYaml . | nindent 8 }}
    {{- end }}
    {{- with .Values.tolerations }}
    tolerations:
      {{- toYaml . | nindent 8 }}
    {{- end }}
```

Explanation:

- **apiVersion**: Specifies the version of the Kubernetes API.
- **kind**: Defines the type of Kubernetes object, in this case, a Deployment.
- **metadata**: Metadata for the Deployment, including its name.
- **spec**: Specification for the Deployment, including replicas, selector, and template.
- **template**: Defines the Pod template, including metadata and spec.
- **containers**: Defines the containers in the Pod, including name, image, ports, environment variables, and volume mounts.

wordpress-chart/templates/service.yaml

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

---

apiVersion: v1
kind: Service
metadata:
  name: mysql
spec:
  selector:
    app: mysql
  ports:
    - protocol: TCP
      port: 3306
      targetPort: 3306
  type: ClusterIP

---
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  type: {{ .Values.service.type }}

---

apiVersion: v1
kind: Service
metadata:
  name: {{ include "wordpress-chart.fullname" . }}
  labels:
    {{- include "wordpress-chart.labels" . | nindent 4 }}
spec:
  type: NodePort  # Change to NodePort
  ports:
    - port: {{ .Values.service.port }}
      targetPort: http
      protocol: TCP
      name: http
  selector:
    {{- include "wordpress-chart.selectorLabels" . | nindent 4 }}

---

apiVersion: v1
kind: Service
metadata:
  name: nginx
spec:
  type: NodePort  # Change to NodePort
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  selector:
    app: nginx
```

Explanation:

- **apiVersion**: Specifies the version of the Kubernetes API.
- **kind**: Defines the type of Kubernetes object, in this case, a Service.
- **metadata**: Metadata for the Service, including its name.
- **spec**: Specification for the Service, including type, ports, and selector.

Step 5: Access the WordPress Application

Get the URL to access the WordPress application:

minikube service my-release-wordpress-chart –url

Open the provided URL in your web browser to access your WordPress site.

Cleanup

To remove the WordPress deployment and free up resources, run:

helm delete my-release

To stop Minikube, run:

minikube stop

Conclusion

This guide provides a detailed walkthrough for deploying a production-grade WordPress application on Kubernetes using Docker and Helm charts. The steps include setting up persistent storage, building and pushing Docker images, creating and deploying Helm charts, and accessing the deployed application.