BookStack Deployment with Longhorn Storage

This document details the deployment of BookStack, a self-hosted, simple knowledge base platform, on a Kubernetes cluster using Docker containers. The deployment utilizes persistent volumes for data persistence and a ConfigMap for configuration.

Overview

This setup deploys BookStack with the following components:

- **BookStack Database:** A MariaDB database container to store BookStack data.
- **BookStack Application:** The BookStack application container serving the web interface.
- **Persistent Volume Claims (PVCs):** Used to request persistent storage for both the database and application data.
- **ConfigMap:** Stores non-sensitive configuration variables for the BookStack application.
- **Secrets:** Stores sensitive information like database passwords and the application key.
- **Ingress:** Configures external access to the BookStack application.

Prerequisites

- A Kubernetes cluster with access to a persistent volume provisioner (e.g., Longhorn).
- `kubectl` configured to interact with your cluster.
- A base64 encoder/decoder utility (e.g., `base64` command line tool).
- Before deploying you might need to disable security until deployment is done and pods are running:

```
kubectl label --overwrite ns bookstack pod-
security.kubernetes.io/enforce=privileged
```

Deployment Steps

Short Version

```
# Create Namespace
kubectl create namespace bookstack

# Apply the different k8s configuration filesin this order
kubectl apply -f bookstack-secrets.yaml
kubectl apply -f bookstack-db-pvc.yaml
kubectl apply -f bookstack-pvc.yaml
kubectl apply -f bookstack-db.yaml
kubectl apply -f bookstack-configmap.yaml
kubectl apply -f bookstack-app.yaml
kubectl apply -f bookstack-ingress.yaml
```

Create Namespace

Create a namespace for the project

```
kubectl create namespace bookstack
```

Configure Secrets

Before applying the `bookstack-secrets.yaml` file, you **must** generate and encode the necessary secrets:

- **Database User:** Choose a username for the BookStack application database user.
- **Database Password:** Create a strong password for the database user.
- **Database Root Password:** Create a strong password for the MariaDB root user.
- **Application Key:** Generate a Laravel application key (e.g., using a Laravel app-key generator) and then base64 encode it.

Use the following command to generate a random base64 encoded string for passwords:

```
openssl rand -hex 16 | base64
```

Example `bookstack-secrets.yaml` (replace placeholders with your encoded values):

```
apiVersion: v1
kind: Secret
metadata:
  name: bookstack-secrets
```

```
  namespace: bookstack
type: Opaque
data:
  db-user: Ym9va3N0YWNr # Replace with base64 encoded username
  db-password: QWJjMTIzIQ== # Replace with base64 encoded password
  db-root-password: cm9vdHBhc3N3b3Jk # Replace with base64 encoded root
password
  app-key:
YmFzZTY0OmZENGJSRCtpU0tXckFDQkMvaTJvUFAybGttbGpBU1RJY2l6UmtRNUhNTjg9 #
Replace with base64 encoded app key
```

Apply the secrets:

```
kubectl apply -f bookstack-secrets.yaml
```

Persistent Volume Claims (PVCs)

The deployment uses two PVCs for persistent storage:

- `bookstack-db-pvc.yaml`: Requests 10Gi of storage for the MariaDB database.
- `bookstack-app-pvc.yaml`: Requests 64Gi of storage for BookStack application configuration and uploaded files.

Apply the PVC definitions:

```
kubectl apply -f bookstack-db-pvc.yaml
kubectl apply -f bookstack-app-pvc.yaml
```

Apply Database and App Deployments including Service

```
kubectl apply -f bookstack-db.yaml
kubectl apply -f bookstack-app.yaml
```

These commands deploy the MariaDB database and the BookStack application. Check the logs to confirm successful database migrations:

```
kubectl -n bookstack logs deployments/bookstack-db
kubectl -n bookstack logs deployments/bookstack
```

Example database log output indicating successful startup:

```
[migrations] started
[migrations] no migrations found
usermod: no changes
...
Connection to localhost (::1) 3306 port [tcp/mysql] succeeded!
Logrotate is enabled
```

Example application log output indicating migration completion:

```
[migrations] started
[migrations] 01-nginx-site-confs-default: executing...
[migrations] 01-nginx-site-confs-default: succeeded ...
```

1. **Apply Ingress:**

```
kubectl apply -f bookstack-ingress.yaml
```

This configures the ingress for BookStack, making it accessible from outside the cluster. This assumes you are using Istio for ingress; adjust as necessary for your ingress controller.

Configuration Details

- **ConfigMap (`bookstack-config.yaml`)**: Provides non-sensitive configuration settings for BookStack, like the application URL, environment, timezone, and session lifetime. The example sets `APP_ENV` to `production` and the session lifetime to 3 days. You can customize these settings as needed.

- **Deployment Strategies:** The BookStack application deployment utilizes a `Recreate` strategy, which terminates the old pod before starting a new one. This is crucial when using `ReadWriteOnce` PVCs to prevent concurrent access to the same volume.

- **Pod Affinity:** The BookStack application deployment includes `podAffinity` to encourage co-location of the application and database pods on the same Kubernetes node. This can reduce network latency.

Best Practices

- **Secrets Management:** Never commit sensitive information directly into your manifests. Use Kubernetes Secrets (as demonstrated) or a dedicated secrets management solution

(e.g., HashiCorp Vault) for production environments.

- **Storage Provisioning:** Ensure you have a suitable persistent volume provisioner configured in your cluster.
- **Resource Limits:** Consider setting resource requests and limits for the containers to ensure stable performance.
- **Monitoring:** Implement monitoring and alerting to track the health and performance of your BookStack deployment.
- **Backup and Recovery:** Establish a backup and recovery strategy for the MariaDB database.

Documentation

- **BookStack Official Documentation:** https://www.bookstackapp.com/docs/

Beispiel Dateien

bookstack-secrets.yaml

```
apiVersion: v1
kind: Secret
metadata:
  name: bookstack-secrets
  namespace: bookstack
type: Opaque
data:
  db-user: <base64 encoded values>
  db-password: <base64 encoded values>
  db-root-password: <base64 encoded values>
  app-key: <base64 encoded values>
```

bookstack-db-pvc.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: bookstack-db-pvc
  namespace: bookstack
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: longhorn
  resources:
```

```
    requests:
      storage: 10Gi
```

bookstack-pvc.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: bookstack-config-pvc
  namespace: bookstack
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: longhorn
  resources:
    requests:
      storage: 64Gi
```

bookstack-db.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: bookstack-db
  namespace: bookstack
spec:
  selector:
    matchLabels:
      app: bookstack-db
  template:
    metadata:
      labels:
        app: bookstack-db
    spec:
      containers:
      - name: mariadb
        image: ghcr.io/linuxserver/mariadb:11.4.9-r0-ls209
        imagePullPolicy: IfNotPresent
        env:
        - name: MARIADB_AUTO_UPGRADE
          value: "1"
        - name: MYSQL_ROOT_PASSWORD
          valueFrom:
            secretKeyRef:
              name: bookstack-secrets
              key: db-root-password
        - name: MYSQL_DATABASE
```

```
          value: "bookstackapp"
        - name: MYSQL_USER
          valueFrom:
            secretKeyRef:
              name: bookstack-secrets
              key: db-user
        - name: MYSQL_PASSWORD
          valueFrom:
            secretKeyRef:
              name: bookstack-secrets
              key: db-password
        ports:
        - containerPort: 3306
        volumeMounts:
        - name: db-data
          mountPath: /var/lib/mysql
      volumes:
      - name: db-data
        persistentVolumeClaim:
          claimName: bookstack-db-pvc

---

apiVersion: v1
kind: Service
metadata:
  name: bookstack-db
  namespace: bookstack
spec:
  selector:
    app: bookstack-db
  ports:
    - port: 3306
```

bookstack-configmap.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: bookstack-env
  namespace: bookstack
data:
  app.env: |
    APP_URL=https://bookstack.example.com
    APP_ENV=production
    APP_LANG=en
    APP_TIMEZONE=Europe/Amsterdam
```

```
    SESSION_LIFETIME=4320
    DISABLE_EXTERNAL_SERVICES=true

    # AUTH_METHOD=ldap
    # LDAP Connection
    # LDAP_SERVER=
    # LDAP_BASE_DN=""


    # The full DN and password of the user used to search the server
    # Can both be left as 'false' (without quotes) to bind anonymously
    # LDAP_DN=""
    # LDAP_PASS="<redacted>"

    # LDAP User Settings / Search Filter
    # LDAP_USER_FILTER=(&(sAMAccountName={user}))

    # LDAP_VERSION=3
    # LDAP_ID_ATTRIBUTE=BIN;objectGUID
    # LDAP_EMAIL_ATTRIBUTE=mail
    # LDAP_DISPLAY_NAME_ATTRIBUTE=cn
    # LDAP_START_TLS=false
    # LDAP_THUMBNAIL_ATTRIBUTE=null
    # LDAP_TLS_INSECURE=true
```

bookstack-app.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: bookstack
  namespace: bookstack
spec:
  replicas: 1  # Single replica since PVC is ReadWriteOnce (Longhorn)
  strategy:
    type: Recreate  # Ensures old pod is terminated before a new one
starts (important for RWO volumes)
  selector:
    matchLabels:
      app: bookstack
  template:
    metadata:
      labels:
        app: bookstack  # Must match selector above
    spec:
      affinity:
        podAffinity:
```

```
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - bookstack-db  # This MUST match the label in the DB
deployment
            topologyKey: "kubernetes.io/hostname"
      containers:
        - name: bookstack
          image:
nexus.amdc2.ncia.nato.int/linuxserver/bookstack:26.03.2
          imagePullPolicy: IfNotPresent  # Avoid unnecessary pulls if
image already exists

          # Loads ALL key/value pairs from the ConfigMap as environment
variables
          # Use this for non-sensitive configuration (APP_URL, LDAP
settings, etc.)
          envFrom:
            - configMapRef:
                name: bookstack-env   # Load values from a config map

          env:
            # Container runtime user mapping (required by linuxserver
images)
            - name: PUID
              value: "1000"
            - name: PGID
              value: "1000"

            # Sensitive values should always come from Secrets
            - name: APP_KEY
              valueFrom:
                secretKeyRef:
                  name: bookstack-secrets
                  key: app-key

            # Reverse proxy handling (trust all proxies)
            - name: APP_PROXIES
              value: "*"

            # Database connection settings
            - name: DB_HOST
              value: "bookstack-db"
            - name: DB_PORT
```

```
              value: "3306"

            # DB credentials (from Secret)
            - name: DB_USERNAME
              valueFrom:
                secretKeyRef:
                  name: bookstack-secrets
                  key: db-user
            - name: DB_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: bookstack-secrets
                  key: db-password

            - name: DB_DATABASE
              value: "bookstackapp"
            - name: APP_URL
              value: "https://bookstack.amdc2.ncia.nato.int"
            - name: APP_ENV
              value: "production"
            - name: APP_LANG
              value: "en"
            - name: APP_TIMEZONE
              value: "Europe/Amsterdam"
            - name: SESSION_LIFETIME
              value: "4320"
            - name: DISABLE_EXTERNAL_SERVICES
              value: "true"

            # NOTE:
            # Any variable defined here OVERRIDES values from envFrom
(ConfigMap)

          resources:
            limits:
              memory: 2048Mi  # Hard cap
            requests:
              cpu: 100m       # Guaranteed CPU
              memory: 1024Mi  # Guaranteed memory

          ports:
            - containerPort: 80  # BookStack runs on port 80 inside
container

          volumeMounts:
            - name: bookstack-config
              mountPath: /config  # Persistent app data (includes
```

```
uploads + app config)

      volumes:
        - name: bookstack-config
          persistentVolumeClaim:
            claimName: bookstack-config-pvc  # Longhorn-backed PVC
(RWO)

---

apiVersion: v1
kind: Service
metadata:
  name: bookstack
  namespace: bookstack
spec:
  selector:
    app: bookstack  # Routes traffic to pods with this label
  ports:
    - protocol: TCP
      port: 80        # Service port (cluster-internal)
      targetPort: 80  # Container port
```

bookstack-ingress.yaml

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: bookstack-routing
  namespace: istio-system
spec:
  hosts:
  - "bookstack.example.com"
  gateways:
  - my-prod-cluster-gateway
  http:
  - route:
    - destination:
        host: bookstack.bookstack.svc.cluster.local
        port:
          number: 80
```

Revision #5

Created 2026-03-27 14:03:24 UTC by Carsten

Updated 2026-03-27 14:53:16 UTC by Carsten