Konfiguration - MediaWiki

- **Verwendung**:
  MediaWiki wird als Container-Applikation im Kubernetes-Cluster betrieben. Die Konfiguration erfolgt über separate YAML-Dateien für Deployment, Netzwerk, Persistenz und Sicherheit.
- **Einsatzgrund**:
  Die Applikation ist ein etabliertes Open-Source-Wiki-System mit vielseitigen Funktionen. Durch den Containerbetrieb wird die Wiederverwendbarkeit, Skalierbarkeit und Portierbarkeit erhöht.
- **Rolle im System**: MediaWiki fungiert als zentrales Intranet der Firma. Es wird über Ingress erreichbar gemacht und nutzt eine persistent gespeicherte MariaDB-Datenbank zur Verwaltung aller Inhalte.

Ressourcen - Anwendung

Im Folgenden sind alle YAML-Dateien aufgeführt, die zur Bereitstellung und Konfiguration der Anwendung benötigt werden. Sie decken u. a. Container-Deployment, Netzwerkzugriff, Speicheranbindung sowie Konfigurations- und Zugriffsdaten ab.

Ressource	Dateiname	Zweck
Deployment	`deployment.yaml`	Startet den App-Container
Service	`service.yaml`	Interner Netzwerkzugriff
Persistent Volume Claim	`pvc.yaml`	Persistenz für Daten
ConfigMap	`configmap.yaml`	Konfigurationsparameter
Secret	`secret.yaml`	Zugangsdaten oder Tokens
Ingress	`mediawiki-ingress.yaml`	Zugriff via Hostname

Ressourcen - Datenbank

Die folgenden YAML-Dateien definieren den Betrieb der zugehörigen Datenbank. Diese nutzt dieselben Konfigurationsdateien wie die Anwendung, um einheitliche und zentrale Umgebungsparameter sicherzustellen.

Ressource	Dateiname	Zweck
Deployment	`db-deployment.yaml`	Startet die Datenbank (`MariaDB`)
Service	`db-service.yaml`	Intern erreichbar durch App
Persistent Volume Claim	`db-pvc.yaml`	Persistenz für Datenbankinhalte
Konfiguration	-	Verwendet dieselbe ConfigMap & Secret wie die Anwendung

Deployment

> Definiert das Deployment für die Anwendung: Container-Image, Umgebungsvariablen, Volumes und Replikation.

deployment.yaml

```yaml
apiVersion: apps/v1                                  # API-Version für Deployments
kind: Deployment                                     # Objekttyp: Deployment
metadata:
  name: mediawiki                                    # Name des Deployments
  namespace: m347-mediawiki                          # Namespace, in dem das
Deployment ausgeführt wird
  labels:
    app: mediawiki                                   # Label zur Gruppierung und
Service-Verknüpfung
spec:
  replicas: 1                                        # Anzahl der gewünschten Pods
(Replikate)
  selector:
    matchLabels:
      app: mediawiki                                 # Verknüpft Selector mit
Template-Label
  template:
    metadata:
      labels:
        app: mediawiki
    spec:
      containers:
        - name: mediawiki                            # Containername
          image: mediawiki:1.43                      # Docker-Image (offizielles
MediaWiki-Image)
          ports:
            - containerPort: 8080                    # Standardport für MediaWiki
          env:                                       # Umgebungsvariablen für
Konfiguration
            - name: MEDIAWIKI_SITE_NAME
              valueFrom:
                configMapKeyRef:
                  name: mediawiki-config
                  key: MEDIAWIKI_SITE_NAME
            - name: MEDIAWIKI_SITE_LANG
              valueFrom:
                configMapKeyRef:
                  name: mediawiki-config
                  key: MEDIAWIKI_SITE_LANG
            - name: MEDIAWIKI_DB_NAME
              valueFrom:
                configMapKeyRef:
                  name: mediawiki-config
                  key: MEDIAWIKI_DB_NAME
            - name: MEDIAWIKI_DB_USER
```

```
            valueFrom:
              configMapKeyRef:
                name: mediawiki-config
                key: MEDIAWIKI_DB_USER
            - name: MEDIAWIKI_DB_HOST
              valueFrom:
                configMapKeyRef:
                  name: mediawiki-config
                  key: MEDIAWIKI_DB_HOST
            - name: MEDIAWIKI_DB_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: mediawiki-secret
                  key: MEDIAWIKI_DB_PASSWORD
          volumeMounts:
            - name: mediawiki-storage
              mountPath: /bitnami/mediawiki     # Mount-Pfad für persistente Daten
      volumes:
        - name: mediawiki-storage
          persistentVolumeClaim:
            claimName: mediawiki-pvc            # Bindung an PVC
```

Erklärung der Konfiguration

- `replicas: 1`
  Es wird **nur ein Pod** instanziiert, ausreichend für Entwicklungsumgebungen.
- `image: mediawiki:1.43`
  Verwendet das offizielle MediaWiki-Docker-Image in der Version 1.43.
- `env`
  Die Konfiguration erfolgt vollständig über Umgebungsvariablen, welche über `ConfigMap` und `Secret` eingebunden werden. Dies fördert die Trennung von Code und Konfiguration.
- `volumeMounts`
  Das MediaWiki-Dateiverzeichnis wird auf `/bitnami/mediawiki` gemountet. Dort werden z. B. Uploads und Erweiterungen gespeichert.
- `persistentVolumeClaim`
  Das Deployment bindet das PVC `mediawiki-pvc`, das in der separaten Datei `pvc.yaml` definiert wurde.

Service

> Stellt einen internen Kubernetes-Service zur Verfügung, über den die App im Cluster erreichbar ist.

`service.yaml`

```
apiVersion: v1                      # Verwendete API-Version für Service
kind: Service                       # Objekttyp: Service (erzeugt interne
Netzwerkschnittstelle)
```

```
metadata:
  name: mediawiki-service           # Interner Name des Service, über den er im
Deployment referenziert wird
  namespace: m347-mediawiki         # Namespace, in dem der Service gespeichert
ist
spec:
  type: ClusterIP                   # Interner Servie (kein externer Zugriff) -
Default
  selector:
    app: mediawiki
  ports:
    - port: 80                      # Service-Port im Cluster
      targetPort: 80                # Weiterleitung an Container-Port (von der
Applikation vorgegeben)
```

Erklärung der Konfiguration

- `type: ClusterIP`
  Dieser Typ ist Standard in Kubernetes und sorgt dafür, dass der Service **nur innerhalb des Clusters erreichbar** ist. Externer Zugriff erfolgt über den **Ingress Controller**.
  → Vorteil: mehr Sicherheit und klare Trennung zwischen internem und externem Verkehr.

- `selector.app: mediawiki`
  Verbindet den Service mit dem **entsprechenden Pod**. Der Selector greift auf das Label `app: mediawiki` im Deployment zurück und stellt so sicher, dass die Netzwerkanfragen an den richtigen Pod weitergeleitet werden.

- `ports.port: 80`
  Der Port, auf dem der Service **innerhalb des Clusters** erreichbar ist.

- `ports.targetPort: 80`
  Der Port, an den der Traffic **im Container selbst** weitergeleitet wird.
  In diesem Fall verwendet der MediaWiki-Container Port 80, da dieser bei Webservern typischerweise als Standard konfiguriert ist.

Persistente Daten (PVC)

> Fordert persistenten Speicher im Cluster an, z. B. für Medien-Uploads oder Logs der Anwendung.

`pvc.yaml`

```
apiVersion: v1                      # Verwendete API-Version für PVC
kind: PersistentVolumeClaim         # Objekttyp: Persistenz (fordert Speicherplatz
beim Cluster an)
metadata:
  name: mediawiki-pvc               # Interner Name des PVC, über den er im
Deployment referenziert wird
  namespace: m347-mediawiki         # Namespace, in dem der PVC gespeichert ist
spec:
  accessModes:
```

```
    - ReadWriteOnce                # Nur ein Node darf gleichzeitig schreiben
  resources:
    requests:
      storage: 10Gi                # Speichergrösse (hier: 10 Gibibyte)
  storageClassName: standard       # Standard Storage-Klasse
```

Erklärung der Konfiguration

- `kind: PersistentVolumeClaim`
  Ein PVC ist eine Anfrage an das Kubernetes-System, Speicherplatz bereitzustellen. In diesem Fall wird ein Volume für **MediaWiki-spezifische Daten** angefordert.

- `name: mediawiki-pvc`
  Dieser Name wird im **Deployment** verwendet, um das Volume korrekt zu mounten.

- `accessModes: ReadWriteOnce`
  Nur ein Node darf das Volume gleichzeitig beschreiben.
  Dies ist der gängigste Modus für Anwendungen, die auf einem einzelnen Pod laufen.

- `resources.requests.storage: 10Gi`
  Reserviert 10 Gibibyte Speicherplatz.
  Die gewählte Grösse basiert auf der Annahme, dass MediaWiki über längere Zeit Medien, Inhalte und Erweiterungen speichern wird.

- `storageClassName: standard`
  Verweist auf die Standard-Storage-Class des Clusters.
  Dies ermöglicht automatische Bereitstellung eines geeigneten `PersistentVolume`.

Datenbank - Deployment

> Startet die zugehörige Datenbankinstanz inkl. Volume, Ports und Konfiguration.

`db-deployment.yaml`

```
apiVersion: apps/v1                                # API-Version für Deployments
kind: Deployment                                   # Objekttyp: Deployment
metadata:
  name: mediawiki-db                               # Name des Deployments
  namespace: m347-mediawiki                        # Namespace, in dem das
Deployment ausgeführt wird
  labels:
    app: mariadb                                   # Label zur Gruppierung und
Service-Verknüpfung
spec:
  replicas: 1                                      # Anzahl der gewünschten Pods
(Replikate)
  selector:
    matchLabels:
      app: mariadb                                 # Verknüpft Selector mit
Template-Label
```

```
  template:
    metadata:
      labels:
        app: mariadb
    spec:
      containers:
        - name: mediawiki-db                          # Containername
          image: bitnami/mariadb:latest               # Docker-Image (verwaltet
durch Bitnami)
          ports:
            - containerPort: 3306                     # Standardport für MariaDB
          env:                                        # Umgebungsvariablen für
Konfiguration
            - name: MARIADB_ROOT_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: mediawiki-secret
                  key: MARIADB_ROOT_PASSWORD
            - name: MARIADB_DATABASE
              valueFrom:
                configMapKeyRef:
                  name: mediawiki-config
                  key: MEDIAWIKI_DB_NAME
            - name: MARIADB_USER
              valueFrom:
                configMapKeyRef:
                  name: mediawiki-config
                  key: MEDIAWIKI_DB_USER
            - name: MARIADB_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: mediawiki-secret
                  key: MEDIAWIKI_DB_PASSWORD
          volumeMounts:
            - name: mediawiki-db-storage
              mountPath: /bitnami/mariadb             # Mount-Pfad für persistente
Daten
      volumes:
        - name: mediawiki-db-storage
          persistentVolumeClaim:
            claimName: mediawiki-db-pvc               # Bindung an PVC
```

Erklärung der Konfiguration

- `image: bitnami/mariadb:latest`
  Verwendet ein vorgefertigtes, sicheres Image mit vorkonfigurierter MariaDB.

- `env`
  Die Datenbankkonfiguration wird über `ConfigMap` und `Secret` eingebunden. Das erlaubt eine flexible Anpassung ohne Änderung des Images.

- `volumeMounts`
  Das Verzeichnis `/bitnami/mariadb` wird verwendet, um Daten persistent abzulegen.

- `persistentVolumeClaim`
  Das Deployment bindet den PVC `mediawiki-db-pvc`, der separat definiert ist.

Datenbank - Service

> Stellt einen internen Kubernetes-Service für die Datenbank bereit, der durch die App genutzt wird.

`db-service.yaml`

```
apiVersion: v1                    # Verwendete API-Version für Service
kind: Service                     # Objekttyp: Service (erzeugt interne
Netzwerkschnittstelle)
metadata:
  name: mediawiki-db-service      # Interner Name des Service, über den er im
Deployment referenziert wird
  namespace: m347-mediawiki       # Namespace, in dem der Service gespeichert ist
spec:
  type: ClusterIP                 # Interner Servie (kein externer Zugriff) -
Default
  selector:
    app: mariadb
  ports:
    - port: 3306                  # Service-Port im Cluster
      targetPort: 3306            # Weiterleitung an Container-Port (von der
Applikation vorgegeben)
```

Erklärung der Konfiguration

- `type: ClusterIP`
  Standardtyp – nur intern erreichbar (kein externer Zugang).

- `selector.app: mariadb`
  Stellt sicher, dass der Service alle Pods mit dem Label `app: mariadb` anspricht (in diesem Fall den MariaDB-Pod).

- `port` und `targetPort`
  Beide auf 3306 gesetzt – dem Standardport für MySQL/MariaDB. Der Service leitet Anfragen direkt an diesen Port im Container weiter.

Datenbank - Persistente Daten (PVC)

> Bindet ein Volume für die dauerhafte Speicherung von Datenbankdaten ein.

`db-pvc.yaml`

```
apiVersion: v1                    # Verwendete API-Version für PVC
kind: PersistentVolumeClaim       # Objekttyp: Persistenz (fordert Speicherplatz
beim Cluster an)
metadata:
  name: mediawiki-db-pvc          # Interner Name des PVC, über den er im
Deployment referenziert wird
  namespace: m347-mediawiki       # Namespace, in dem der PVC gespeichert ist
spec:
  accessModes:
    - ReadWriteOnce               # Nur ein Node darf gleichzeitig schreiben
  resources:
    requests:
      storage: 1Gi                # Speichergrösse (hier: 1 Gibibyte)
```

Erklärung der Konfiguration

- `accessModes: ReadWriteOnce`
  Das Volume kann nur von einem Pod gleichzeitig beschrieben werden – ideal für Einzel-Instanzen.
- `resources.requests.storage: 1Gi`
  Reserviert 1 Gibibyte Speicherplatz. Für Testszenarien und kleinere Wikis ist dies ausreichend.
- `namespace: m347-mediawiki`
  Der PVC ist im gleichen Namespace wie das Deployment (`mediawiki-db`), was zwingend notwendig ist, damit die Zuordnung funktioniert.

ConfigMap & Secret

> Definiert zentrale Konfigurationswerte (ConfigMap) und vertrauliche Daten (Secret), die in App und DB referenziert werden.

ConfigMap - `configmap.yaml`

```
apiVersion: v1                                # Verwendete API-Version für ConfigMap
kind: ConfigMap                               # Objekttyp: Konfigurationsdaten
metadata:
  name: mediawiki-config                      # Interner Name der ConfigMap, über
den sie im Deployment referenziert wird
  namespace: m347-mediawiki                   # Namespace, in dem die ConfigMap
gespeichert ist
data:                                         # Ab hier: Schlüssel-Wert-Paare
  MEDIAWIKI_SITE_NAME: Intranet
  MEDIAWIKI_SITE_LANG: de
  MEDIAWIKI_DB_NAME: mediawiki
  MEDIAWIKI_DB_USER: mwuser
  MEDIAWIKI_DB_HOST: mediawiki-db-service
```

ConfigMap - Erklärung der Konfiguration

Die `ConfigMap` enthält zentrale, **nicht-sensitive Konfigurationswerte** für die MediaWiki-Instanz. Diese Werte werden in den Deployments der App- und Datenbank-Container als Umgebungsvariablen eingebunden.

Begründung:

- Durch die Auslagerung in eine `ConfigMap` können Konfigurationswerte unabhängig vom Containerimage definiert und bei Bedarf zentral angepasst werden.
- Sensible Daten (wie Passwörter) werden **nicht** hier, sondern im `Secret` verwaltet, was die Sicherheit erhöht.

Erklärung der Schlüssel

Schlüssel	Beschreibung
`MEDIAWIKI_SITE_NAME`	Legt den Namen der MediaWiki-Installation fest, wird z. B. in der Titelzeile angezeigt.
`MEDIAWIKI_SITE_LANG`	Bestimmt die Standardsprache der Benutzeroberfläche.
`MEDIAWIKI_DB_NAME`	Gibt an, wie die Datenbank innerhalb von MariaDB heisst. Muss mit dem DB-Setup übereinstimmen.
`MEDIAWIKI_DB_USER`	Datenbankbenutzer, mit dem sich MediaWiki mit der Datenbank verbindet.
`MEDIAWIKI_DB_HOST`	Interner DNS-Name des Datenbankservices im Cluster. Muss exakt dem Namen des MariaDB-Services entsprechen.

Secret - `secret.yaml`

```
apiVersion: v1                                  # Verwendete API-Version für Secret
kind: Secret                                    # Objekttyp: Secret (Zugangsdaten /
Token)
metadata:
  name: mediawiki-secret                        # Interner Name der Secret, über den
sie im Deployment referenziert wird
  namespace: m347-mediawiki                     # Namespace, in dem die Secret
gespeichert ist
type: Opaque                                    # Standardtyp für benutzerdefinierte
Schlüssel-Wert-Paare
stringData:                                     # Ab hier: Schlüssel-Wert-Paare
  MARIADB_ROOT_PASSWORD: supergeheim            # Diese können in Klartext angegeben
werden (werden intern automatisch Base64-kodiert)
  MEDIAWIKI_DB_PASSWORD: mwpass123
```

Secret - Erklärung der Konfiguration

Die `Secret` enthält **sensible Zugangsdaten**, die für den Betrieb der MediaWiki-Instanz notwendig sind - insbesondere zur Authentifizierung bei der MariaDB-Datenbank. Sie wird im Deployment von App- und Datenbank-Container referenziert.

Begründung:

- Secrets werden verwendet, um vertrauliche Informationen wie Passwörter, Tokens oder API-Keys **verschlüsselt im Cluster** zu speichern.
- Die Werte unter `stringData` können **leserlich im YAML** definiert werden - Kubernetes wandelt sie beim Speichern automatisch in Base64-kodierte Einträge um.
- Der Typ `Opaque` ist der Standardtyp für frei definierte Schlüssel-Wert-Paare.

Erklärung der Schlüssel

Schlüssel	Beschreibung
`MARIADB_ROOT_PASSWORD`	Root-Passwort für die MariaDB-Instanz, erforderlich beim Start der Datenbank.
`MEDIAWIKI_DB_PASSWORD`	Passwort des spezifischen Datenbank-users (`mwuser`), mit dem sich MediaWiki anmeldet.

> [!NOTE] **Best Practices im produktiven Betrieb**:
>
> - Secrets sollten **nicht versioniert** oder öffentlich gespeichert werden.
> - Der Zugriff auf Secrets sollte im Cluster über **Rollen & Rechte** (RBAC) eingeschränkt werden.

Ingress / Externer Zugriff

> Regelt den externen Zugriff auf die Anwendung über Hostnamen mithilfe eines Ingress Controllers.

Die Datei `mediawiki-ingress.yaml` definiert, unter welchem Hostnamen (`mediawiki.m347.ch`) die MediaWiki-Applikation von ausserhalb des Clusters erreichbar ist.
Sie verweist auf den zentralen Ingress Controller und sorgt für die Weiterleitung eingehender Anfragen an den zugehörigen Service der MediaWiki-Anwendung.
Da das zugrundeliegende Ingress-System für alle Anwendungen identisch ist, wird die übergeordnete Konfiguration des Ingress Controllers inklusive Routingprinzipien und Klassendefinition zentral in der Konfigurationsdatei des Ingress Controllers dokumentiert.

Besonderheiten & Herausforderungen

Eine zentrale Herausforderung bei der Bereitstellung von MediaWiki war die **nicht persistente Speicherung des Verzeichnisses** `/var/www/html/` im offiziellen Docker-Image. Obwohl die Datei `LocalSettings.php` korrekt im persistenten Volume (`/bitnami/mediawiki/`) abgelegt wurde, erkannte MediaWiki diese nicht – da sie zusätzlich im nicht-persistenten Webroot (`/var/www/html/`) vorhanden sein muss.

Dieses Verhalten war uns zunächst nicht bekannt und wurde erst durch gezieltes Testen und Analyse entdeckt. **Die Konsequenz**: Nach jedem Pod-Neustart muss `LocalSettings.php` manuell erneut an `/var/www/html/` kopiert werden, um MediaWiki korrekt zu starten. Dieses Vorgehen wurde als Workaround dokumentiert und in den Testplan aufgenommen.

> [!TIP] Für eine produktive Umgebung wäre ein zusätzliches Init-Skript denkbar, das `LocalSettings.php` bei jedem Start automatisch an den richtigen Ort kopiert.