Konfiguration - Redmine

- **Verwendung**:
  Redmine wird als Container-Applikation innerhalb eines Kubernetes-Clusters betrieben. Die Konfiguration erfolgt über eigenständige YAML-Dateien, die das Deployment der Applikation, den Netzwerkzugriff, persistente Speicherung und sichere Verwaltung sensibler Daten definieren.

- **Einsatzgrund**:
  Redmine ist eine etablierte Open-Source-Projektmanagement-Plattform, die besonders für Issue-Tracking, Projektverwaltung und Wissensmanagement genutzt wird. Der Betrieb als Container ermöglicht flexible Skalierung, einfache Portabilität und schnelle Wiederherstellbarkeit der Anwendung.

- **Rolle im System**:
  Redmine dient als zentrale Plattform zur Verwaltung von Projekten, Tickets, Dokumentationen und zur internen Kommunikation. Die Anwendung ist extern über Ingress erreichbar und verwendet eine persistente PostgreSQL-Datenbank.

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	`redmine-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 (`PostgreSQL`)
Service	`db-service.yaml`	Intern erreichbar durch App

Ressource	Dateiname	Zweck
Persistent Volume Claim	`db-pvc.yaml`	Persistenz für Datenbankinhalte
ConfigMap	`db-configmap.yaml`	Definiert z. B. den Datenbanknamen, Benutzername (nicht-sensibel)
Secret	`db-secret.yaml`	Speichert sensible Zugangsdaten wie Passwort verschlüsselt

Files

Deployment

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

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redmine
  namespace: m347-redmine
spec:
  replicas: 1
  selector:
    matchLabels:
      app: redmine
  template:
    metadata:
      labels:
        app: redmine
    spec:
      containers:
      - name: redmine
        image: redmine:latest
        ports:
        - containerPort: 3000
        envFrom:
        - configMapRef:
            name: redmine-config
        - secretRef:
            name: redmine-secret
        volumeMounts:
        - name: redmine-data
          mountPath: /usr/src/redmine/files
      volumes:
      - name: redmine-data
        persistentVolumeClaim:
          claimName: redmine-pvc
```

Erklärung der Konfiguration (Redmine Deployment)

- **replicas: 1**
  Erstellt **einen Pod**, ausreichend für Entwicklungs- oder Testumgebungen.
- **image: redmine:latest**
  Verwendet das offizielle Docker-Image von Redmine in der neuesten Version.
- **ports** Der Container ist auf Port `3000` erreichbar, der Standard-Port von Redmine.
- **envFrom**
  Umgebungsvariablen für Redmine werden über externe Konfigurationsressourcen geladen:
  - **configMapRef**: Lädt Werte aus einer ConfigMap (`redmine-config`).
  - **secretRef**: Lädt vertrauliche Werte aus einem Secret (`redmine-secret`).
- **volumeMounts**
  Mountet persistenten Speicher in den Containerpfad `/usr/src/redmine/files`, um hochgeladene Dateien dauerhaft zu speichern.
- **persistentVolumeClaim**
  Verwendet einen PersistentVolumeClaim (`redmine-pvc`) zur dauerhaften Datenspeicherung, definiert in einer separaten PVC-YAML-Datei.

Service

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

```yaml
apiVersion: v1
kind: Service
metadata:
  name: redmine-service
  namespace: m347-redmine
spec:
  type: ClusterIP
  selector:
    app: redmine
  ports:
  - protocol: TCP
    port: 80
    targetPort: 3000
```

Erklärung der Konfiguration (Redmine Service)

- **kind: Service**
  Erstellt einen internen Kubernetes-Service, der Redmine innerhalb des Clusters erreichbar macht.
- **type: ClusterIP**
  Der Service ist ausschliesslich innerhalb des Kubernetes-Clusters erreichbar und hat keine externe IP-Adresse.

- **selector (app: redmine)**
  Wählt Pods mit dem Label `app: redmine`, um Anfragen an den korrekten Pod weiterzuleiten.
- **ports**
  - **protocol: TCP**: Der Service nutzt das TCP-Protokoll.
  - **port: 80**: Der Port, auf dem der Service innerhalb des Clusters erreichbar ist.
  - **targetPort: 3000**: Der Port des Redmine-Containers, auf dem die Anwendung tatsächlich läuft.

Persistente Daten (PVC)

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

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: redmine-pvc
  namespace: m347-redmine
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```

Erklärung der Konfiguration (Redmine PersistentVolumeClaim)

- **kind: PersistentVolumeClaim**
  Fordert persistenten Speicherplatz zur dauerhaften Speicherung von Anwendungsdaten (z.B. Uploads oder Logfiles) an.
- **accessModes: ReadWriteOnce**
  Der Speicherplatz ist für genau einen Pod gleichzeitig beschreibbar.
- **resources.requests.storage: 10Gi**
  Fordert 10 Gibibyte Speicherplatz an, geeignet für Entwicklungs- und Testumgebungen.
- **storageClassName**
  Nutzt die Standard-Storage-Class des Kubernetes-Clusters (implizit), die definiert, wie und wo der Speicher bereitgestellt wird.

Datenbank - Deployment

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

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redmine-postgres
```

```
  namespace: m347-redmine
spec:
  replicas: 1
  selector:
    matchLabels:
      app: redmine-postgres
  template:
    metadata:
      labels:
        app: redmine-postgres
    spec:
      containers:
      - name: postgres
        image: postgres:15
        ports:
        - containerPort: 5432
        envFrom:
        - configMapRef:
            name: redmine-postgres-config
        - secretRef:
            name: redmine-postgres-secret
        volumeMounts:
        - name: postgres-data
          mountPath: /var/lib/postgresql/data
      volumes:
      - name: postgres-data
        persistentVolumeClaim:
          claimName: redmine-postgres-pvc
```

Erklärung der Konfiguration (Redmine PostgreSQL Deployment)

- **replicas: 1**
  Erstellt **einen Pod**, ausreichend für Entwicklungs- und Testzwecke.
- **image: postgres:15**
  Verwendet das offizielle Docker-Image von PostgreSQL in Version 15.
- **ports** Der Container stellt den Standard-Port `5432` für PostgreSQL bereit.
- **envFrom** Lädt Umgebungsvariablen aus externen Konfigurationsquellen:
  - **configMapRef**: Werte werden aus der ConfigMap (`redmine-postgres-config`) geladen.
  - **secretRef**: Vertrauliche Werte werden aus einem Secret (`redmine-postgres-secret`) bezogen.
- **volumeMounts** Persistenter Speicher wird in den Containerpfad `/var/lib/postgresql/data` eingebunden, um Datenbankdaten dauerhaft zu speichern.
- **persistentVolumeClaim** Nutzt ein PersistentVolumeClaim (`redmine-postgres-pvc`) für die langfristige Speicherung von Datenbankinhalten, definiert in einer separaten PVC-YAML.

Datenbank - Service

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

```
apiVersion: v1
kind: Service
metadata:
  name: postgres
  namespace: m347-redmine
spec:
  selector:
    app: redmine-postgres
  ports:
  - protocol: TCP
    port: 5432
    targetPort: 5432
```

Erklärung der Konfiguration (Redmine PostgreSQL Service)

- **kind: Service**
  Erstellt einen Kubernetes-internen Service, der PostgreSQL im Cluster bereitstellt.
- **selector (app: redmine-postgres)**
  Der Service wählt Pods mit dem Label `app: redmine-postgres` aus, um Datenbankanfragen korrekt weiterzuleiten.
- **ports**
  - **protocol: TCP**: Die Kommunikation erfolgt über TCP.
  - **port: 5432**: Der Port, auf dem PostgreSQL innerhalb des Clusters erreichbar ist.
  - **targetPort: 5432**: Der Container-Port, auf dem PostgreSQL tatsächlich läuft.

Datenbank - Persistente Daten (PVC)

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

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: redmine-postgres-pvc
  namespace: m347-redmine
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi
```

Erklärung der Konfiguration (Redmine PostgreSQL PersistentVolumeClaim)

- **kind: PersistentVolumeClaim**
  Fordert persistenten Speicherplatz für die dauerhafte Speicherung von PostgreSQL-Datenbankdaten an.
- **accessModes: ReadWriteOnce**
  Der Speicherplatz kann gleichzeitig nur von einem Pod beschreibbar genutzt werden.
- **resources.requests.storage: 5Gi**
  Fordert 5 Gibibyte Speicherplatz an, ausreichend für kleinere oder Test-Datenbanken.
- **storageClassName**
  Nutzt die Standard-Storage-Class des Kubernetes-Clusters (implizit), welche festlegt, wie der Speicher bereitgestellt wird.

ConfigMap & Secret

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

ConfigMap

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: redmine-postgres-config
  namespace: m347-redmine
data:
  POSTGRES_DB: redmine_db
  POSTGRES_USER: redmineuser
```

Erklärung der Konfiguration (Redmine PostgreSQL ConfigMap)

- **kind: ConfigMap**
  Definiert zentrale, nicht-vertrauliche Konfigurationswerte für PostgreSQL.
- **data**
  - **POSTGRES_DB: redmine_db**
    Name der PostgreSQL-Datenbank, welche von Redmine verwendet wird.
  - **POSTGRES_USER: redmineuser**
    Benutzername für den PostgreSQL-Datenbankzugriff.

Datenbank - Configmap

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: redmine-postgres-config
  namespace: m347-redmine
```

```
data:
  POSTGRES_DB: redmine_db
  POSTGRES_USER: redmineuser
```

Erklärung der Konfiguration (Redmine PostgreSQL ConfigMap)

- **kind: ConfigMap**
  Enthält zentrale, nicht-vertrauliche Konfigurationswerte für die PostgreSQL-Datenbankinstanz.
- **data**
  - **POSTGRES_DB**
    Legt den Namen (`redmine_db`) der PostgreSQL-Datenbank fest, die von Redmine verwendet wird.
  - **POSTGRES_USER**
    Definiert den Benutzernamen (`redmineuser`) zur Authentifizierung an der PostgreSQL-Datenbank.

Secret

```
apiVersion: v1
kind: Secret
metadata:
  name: redmine-secret
  namespace: m347-redmine
type: Opaque
data:
  REDMINE_DB_PASSWORD: cmVkbWluZXBhc3M=  # redminepass (base64 codiert da
Kubernetes verlangt dass Passwörter und Sensible Daten codiert abgelegt werden)
```

Erklärung der Konfiguration (Redmine Secret)

- **kind: Secret**
  Enthält vertrauliche Daten, wie z.B. Passwörter, in verschlüsselter Form.
- **type: Opaque**
  Ein generisches Kubernetes Secret, welches beliebige vertrauliche Daten speichert.
- **data**
  - **REDMINE_DB_PASSWORD**
    Base64-kodiertes Passwort für den Redmine-Datenbankbenutzer.
    (Kodierung erforderlich, da Kubernetes Passwörter ausschliesslich verschlüsselt speichert.)

Datenbank - Secret

```
apiVersion: v1
kind: Secret
metadata:
  name: redmine-postgres-secret
  namespace: m347-redmine
type: Opaque
data:
  POSTGRES_PASSWORD: cmVkbWluZXBhc3M=  # redminepass (base64 codiert da Kubernetes verlangt dass Passwörter und Sensible Daten codiert abgelegt werden)
```

Erklärung der Konfiguration (Redmine PostgreSQL Secret)

- **kind: Secret**
  Enthält vertrauliche Daten, wie Passwörter, in verschlüsselter Form.
- **type: Opaque**
  Ein generisches Kubernetes Secret, das beliebige vertrauliche Daten speichern kann.
- **data**
  - **POSTGRES_PASSWORD**
    Base64-kodiertes Passwort (`redminepass`) für den PostgreSQL-Datenbankbenutzer.
    (Kodierung erforderlich, da Kubernetes Passwörter ausschliesslich verschlüsselt speichert.)

Ingress / Externer Zugriff

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

Die Datei `redmine-ingress.yaml` definiert, unter welchem Hostnamen (`redmine.m347.ch`) die Redmine-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 Redmine-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

Die Nutzung von Redmine innerhalb eines Kubernetes-Clusters hat besonders die Bedeutung von korrekter Netzwerkkommunikation und Datenpersistenz hervorgehoben. Eine wesentliche Herausforderung lag darin, die Interaktion zwischen Redmine und der MariaDB-Datenbank korrekt und sicher zu gestalten. Dies erforderte eine präzise Konfiguration der Zugriffsrechte, sorgfältiges Management der persistenten Speicherlösungen und eine exakte Abstimmung zwischen Applikation und Datenbankdiensten. Besonders die Verwaltung sensibler Informationen über Secrets und deren sichere Einbindung in den Deployment-Prozess stellte eine wichtige Aufgabe dar, die sorgfältiges Vorgehen und umfassendes Verständnis der Sicherheitsmechanismen von Kubernetes verlangte.