Konfiguration - Ingress Controller

- **Verwendung**:
  Ingress wird eingesetzt, um den **externen Zugriff** auf alle Webanwendungen im Cluster über Hostnamen wie `mediawiki.m347.ch` oder `grafana.m347.ch` bereitzustellen.
- **Einsatzgrund**:
  Statt jede Anwendung einzeln per NodePort zugänglich zu machen, ermöglicht Ingress einen **zentralen Einstiegspunkt** mit klarer Trennung zwischen internem und externem Netzwerk.
- **Rolle im System**:
  Der Ingress Controller fungiert als **Reverse Proxy**, der eingehende HTTP-Anfragen entgegennimmt und anhand der konfigurierten Regeln an den passenden Kubernetes-Service weiterleitet.

Ressourcen - Anwendung

Im Folgenden ist die zentrale Ressource aufgeführt, welche die Ingress-Klasse und Konfiguration des Controllers definiert.

Ressource	Dateiname	Zweck
IngressClass & Deployment	`ingress-class.yaml`	Installiert und konfiguriert den Ingress Controller
Ingress-Routen	`[name]-ingress.yaml`	Weiterleitung von externem Traffic an den jeweiligen Service

Files

IngressClass & Controller-Deployment

> Installiert den Ingress Controller im Cluster (z. B. mit nginx) und registriert ihn unter einer benannten Klasse. Diese Klasse wird später von den einzelnen Ingress-Ressourcen referenziert.

- Die IngressClass wird benötigt, um zwischen verschiedenen Ingress-Controllern zu unterscheiden.
- In unserem Fall kommt der weit verbreitete `nginx`-Controller zum Einsatz, welcher unter `k8s.io/ingress-nginx` geführt wird.
- Diese Datei wird nur **einmal zentral** benötigt.

```
apiVersion: networking.k8s.io/v1        # API-Version für IngressClass
kind: IngressClass                      # Objekttyp: Ingress-Klasse
metadata:
  name: nginx                           # Name der Klasse
spec:
  controller: k8s.io/ingress-nginx      # Controller, der diese Klasse verwalten
soll
```

Ingress-Routen

> Leiten eingehenden Webtraffic anhand des Hostnamens an den entsprechenden Service im Cluster weiter.

Die YAML-Struktur basiert auf dem Beispiel, das bei der Installation von `ingress-nginx` via Helm im Terminal ausgegeben wird – dieses wurde projektweit als Vorlage verwendet.

- Jede App erhält ein Ingress-Objekt mit eindeutiger Subdomain.
- `ingressClassName` verbindet die Regel mit dem richtigen Controller.
- `service.name` verweist auf den zugehörigen Kubernetes-Service innerhalb des Namespaces.
- `path: /` sorgt dafür, dass alle Anfragen auf der Root-URL direkt an die App weitergeleitet werden.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: [name]-ingress                          # Name des Ingress-Objekts
  namespace: m347-[name]                        # Namespace, in dem der dieser
Ingress gespeichert ist
spec:
  ingressClassName: nginx
  rules:
    - host: [name].m347.ch                      # Hostname (für den lokalen Zugriff)
      http:
        paths:
          - path: /
            pathType: ImplementationSpecific
            backend:
              service:
                name: [name]-service            # Name des Ziel-Services
                port:
                  number: 80
```

Besonderheiten & Herausforderungen

Eine zentrale Besonderheit beim Einsatz des Ingress Controllers ist, dass der Zugriff über definierte Hostnamen (z. B. `mediawiki.m347.ch`) **nur bei aktivem** `minikube tunnel` funktioniert. Dieser Tunnel ist notwendig, um externe Anfragen korrekt an den Cluster weiterzuleiten, andernfalls bleiben die Dienste von aussen nicht erreichbar.