K8sIngressService

September 4, 2026

K8sIngressService

Kind: Service

Source: `atloria-monorepo/apps/api/src/project/k8s-ingress.service.ts`

`K8sIngressService` manages Kubernetes Ingress resources for a project, including creating or updating ingress configuration, checking TLS certificate readiness, and removing ingress resources when they are no longer needed. It also exposes the hosts currently managed by the service so other backend workflows can coordinate DNS, deployment status, and cleanup.

Methods

Method	Signature	Returns	Description
`ensureIngress`	`ensureIngress(domain: string)`	`Promise<{ ok: boolean; secretName: string; error?: string }>`	Create (or leave in place) the Ingress for a verified domain.
`certReady`	`certReady(domain: string)`	`Promise<boolean>`	Is the certificate issued?
`removeIngress`	`removeIngress(domain: string)`	`Promise<void>`	Remove the Ingress + TLS secret when a domain is deleted.
`listManagedHosts`	`listManagedHosts()`	`Promise<string[]	null>`	

Where it refuses work

- `K8sIngressService` stops the work with an early return when `!this.available`, in 3 places.

- `K8sIngressService` stops the work with an early return when `res.status !== 200`, in 2 places.
- `K8sIngressService` stops the work with an early return when `create.status === 409`.

When something fails

- `K8sIngressService` handles failure in 5 places: it turns it into a return value in 3, logs it and continues in 1, and discards it silently in 1. A failure discarded silently leaves no trace for whoever debugs this later.

Diagram

```
sequenceDiagram
  participant Caller as Project Workflow
  participant Service as K8sIngressService
  participant K8s as Kubernetes API
  participant Cert as Certificate Controller

  Caller->>Service: ensureIngress()
  Service->>K8s: Create or update Ingress
  K8s->>Cert: Request/provision TLS certificate
  Service-->>Caller: { ok, secretName, error? }

  Caller->>Service: certReady()
  Service->>K8s: Read certificate/secret status
  K8s-->>Service: Certificate state
  Service-->>Caller: true / false

  Caller->>Service: listManagedHosts()
  Service->>K8s: Read managed Ingress hosts
  K8s-->>Service: Host rules
  Service-->>Caller: string[] | null

  Caller->>Service: removeIngress()
  Service->>K8s: Delete managed Ingress
  K8s-->>Service: Deletion complete
```

Usage

```
import { Injectable } from '@nestjs/common';
import { K8sIngressService } from './k8s-ingress.service';

@Injectable()
export class ProjectDeploymentService {
  constructor(private readonly ingressService: K8sIngressService) {}
```

```
  async provisionPublicEndpoint() {
    const result = await this.ingressService.ensureIngress();

    if (!result.ok) {
      throw new Error(
        `Failed to provision ingress: ${result.error ?? 'Unknown error'}`,
      );
    }

    const certificateReady = await this.ingressService.certReady();

    return {
      tlsSecretName: result.secretName,
      certificateReady,
      managedHosts: await this.ingressService.listManagedHosts(),
    };
  }

  async teardownPublicEndpoint(): Promise<void> {
    await this.ingressService.removeIngress();
  }
}
```

AI Coding Instructions

- Call `ensureIngress()` before checking certificate status; use its `ok` flag and `error` value to handle provisioning failures explicitly.
- Treat `certReady()` as an asynchronous readiness check, not a guarantee that the ingress endpoint is immediately reachable.
- Handle `listManagedHosts()` returning `null`, which may indicate that no managed ingress or host configuration is available.
- Use `removeIngress()` during project teardown or domain cleanup flows to avoid leaving orphaned Kubernetes resources.
- Keep Kubernetes resource naming, labels, annotations, and host ownership consistent with the conventions already used by this service.

Referenced By

- `ProjectDomainsService` (DEPENDS_ON)
- `ProjectModule` (MODULE_PROVIDES)