NGINX
Part of F5

A Guide to Choosing an Ingress Controller

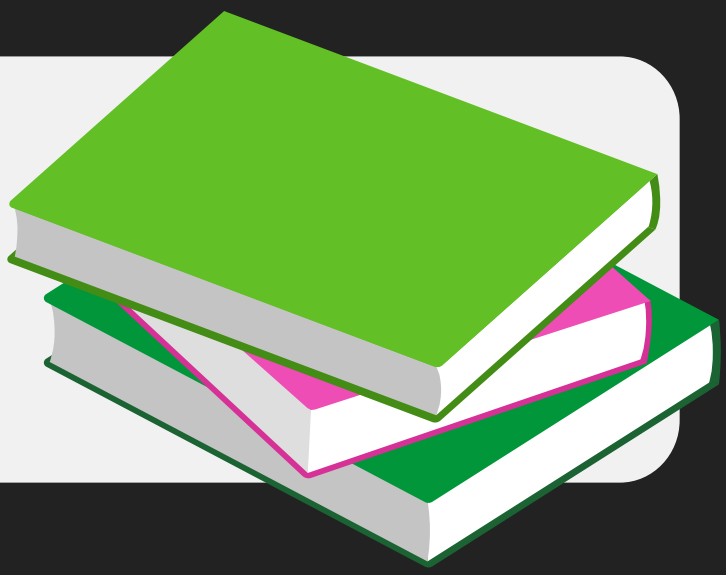
Make the “Promise” of Kubernetes a Reality

- The **promise** of Kubernetes is that organizations can deliver outstanding digital experiences **faster** and **more securely** while **lowering costs**.

- But whether you’re just beginning a cloud migration or are already a microservices expert, you probably know that operating a production Kubernetes system is **hard**. In fact, often Kubernetes makes it more difficult to **secure**, **understand**, and **see** your apps.

- An **Ingress controller** can be one of the most **powerful tools** in your Kubernetes stack – helping you make this “promise” a reality.

Read on to learn the basics on Ingress controllers and how to make a **wise choice** that delivers the functionality and security you need, **today** and **tomorrow**.



What Does An Ingress Controller Do?

The **Ingress controller** is a specialized load balancer that manages Layer 4 and 7 ingress and egress ("north-south") traffic.

It can also be used for:

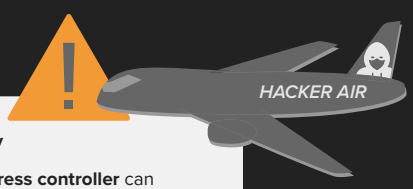
- Traffic control
- Traffic shaping
- Monitoring and visibility
- As an API gateway
- Authentication and SSO
- WAF integration

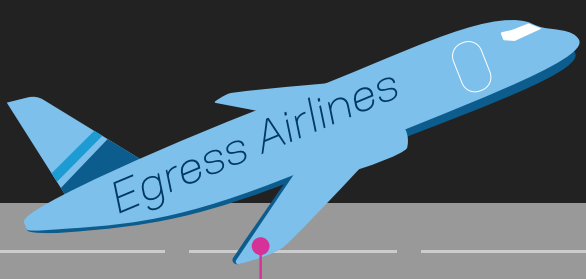
Monitoring and Visibility

The **Ingress controller** can give you insight into issues impacting app and infrastructure performance, and help you predict when traffic surges will strike.

Security

The **Ingress controller** can protect your environment from unauthorized or malicious traffic via centralized authentication, single-sign on (SSO), and as the ideal point for a web application firewall (WAF).



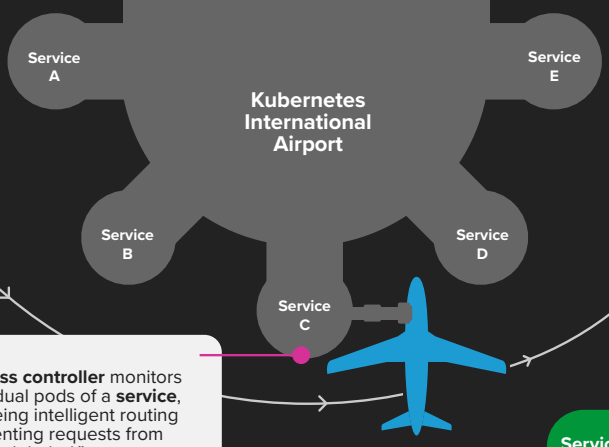


Ingress traffic is traffic entering a Kubernetes cluster.

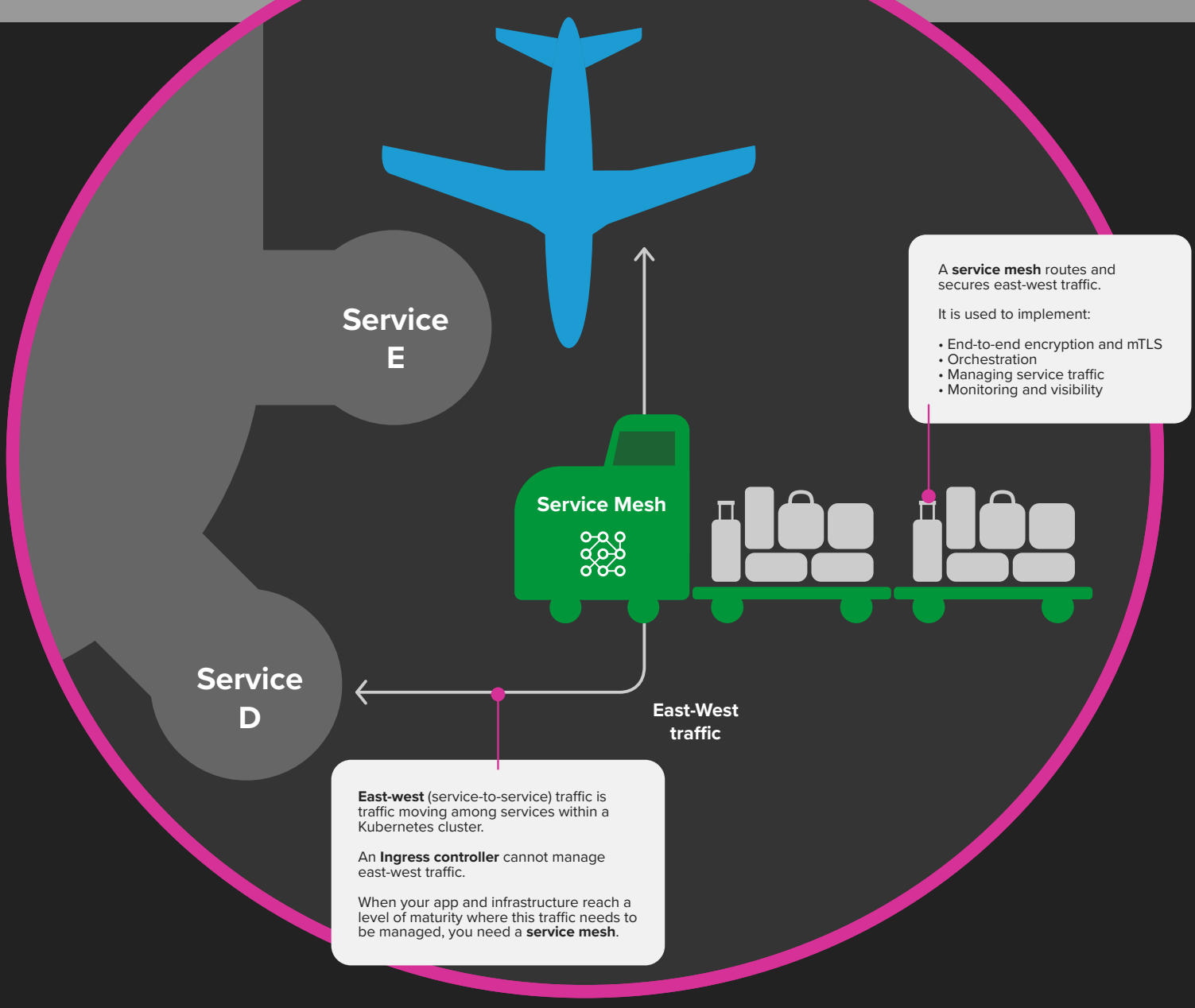
The **Ingress controller** accepts ingress traffic, potentially modifies (shapes) it, and distributes it to pods running inside the environment.

Egress traffic is traffic exiting a Kubernetes cluster.

The **Ingress controller** implements egress rules to enhance security with mutual TLS (mTLS) or limits outgoing traffic from certain pods to specific external services.



The **Ingress controller** monitors the individual pods of a **service**, guaranteeing intelligent routing and preventing requests from being "black-holed."



A **service mesh** routes and secures east-west traffic.

It is used to implement:

- End-to-end encryption and mTLS
- Orchestration
- Managing service traffic
- Monitoring and visibility

East-west (service-to-service) traffic is traffic moving among services within a Kubernetes cluster.

An **Ingress controller** cannot manage east-west traffic.

When your app and infrastructure reach a level of maturity where this traffic needs to be managed, you need a **service mesh**.

How Are You Going to Resource the Ingress Controller?

Budgeting for Capital Costs

Budgeting for Time Costs

Ingress Controller Risks

New tools can introduce risks that might outweigh the rewards. Here are the top three risks that can be introduced by an Ingress controller that doesn't align to your needs.

01 Complexity

Does it Defeat the Purpose of a Microservices Architecture?

Complexity is one of the top challenges in using and deploying containers.[1]

The wrong Ingress controller can add even more complexity – limiting your ability to scale the deployment horizontally and negatively impacting app performance.

02 Latency

Does the Ingress Controller Slow Down Your Apps?

Organizations adopt Kubernetes for the ability to deploy new apps more quickly.[2]

But an Ingress controller that adds latency through errors, timeouts, and reloads can slow down your apps.

03 Security

Does the Ingress Controller Open the Door for Hackers?

More than half of organizations have delayed or slowed down application deployment into production due to container or Kubernetes security concerns.[3]

Watch out for Ingress controllers with slow CVE patching and beware of relying on support from public forums.

[1] CNCF Survey 2020
[2] 2021 Kubernetes Adoption Survey
[3] Red Hat State of Kubernetes Security Report

Future-Proof Your Ingress Controller

Even if you're just starting to dabble in Kubernetes, there's a good chance you aspire to put it into production someday.

There are four main areas where your needs are likely to grow over time.

01

Infrastructure

Will You Use Kubernetes in Hybrid- or Multi-Cloud Environments?

It's rare for an organization to be fully and permanently committed to one type of environment. Choose an infrastructure-agnostic Ingress controller from the start, allowing you to use the same tool across all your environments.

02

Security

How Will You Secure Kubernetes from the Inside?

Kubernetes apps are best protected when security – including authentication and authorization – is close to the apps. Centralizing security (authentication, authorization, DoS protection, web application firewall) at the point of Ingress makes a lot of sense from the standpoint of both cost and efficiency.

03

Support

How "On Your Own" Can You Afford to Be?

Workaround and waiting on community support is okay when you're running small deployments but it's not sustainable when you move to production. Choose an Ingress controller that allows you to add support in the future – or have an inexpensive support tier that can be upgraded as you scale.

04

Multi-Tenancy

How Can Multiple Teams and Apps Share a Container Environment Safely and Securely?

When your services and teams grow in size and complexity, you'll probably turn to multi-tenancy to achieve maximum efficiency. Some Ingress controllers can help you carve up those clusters through a number of features and concepts: multiple ingresses, classes, namespaces, and scoped resources that support setting role-based access controls (RBAC).

Open Source Ingress Controllers

Maintained by a community of users and volunteer developers, though some also have dedicated engineering teams.

Pros

Top reasons an open source Ingress controller could be right for you.

- ▲ **No Monetary Investment (Free!)**
- ▲ **Community-Driven**
- ▲ **High Feature Velocity**

Ideal when...
You're just getting started in Kubernetes, in testing, or low-volume production.

Cons

Top reasons an open source Ingress controller could be wrong for you.

- ▼ **Costs More of Your Time**
- ▼ **Risks of Instability, Insecurity, Unreliability**
- ▼ **Minimal or No Support**

Consider "default" or "commercial" options to outweigh these cons.

Default Ingress Controllers

Developed and maintained by a company that provides a full Kubernetes platform (and often support in managing it).

Pros

Top reasons a default Ingress controller could be right for you

- ▲ **Free or Low Cost**
- ▲ **Reliable**
- ▲ **Supported**

Ideal when...
You're using a Kubernetes platform and are just getting started, in testing, or low-volume production.

Cons

Top reasons a default Ingress controller could be wrong for you

- ▼ **Infrastructure Lock-In**
- ▼ **Basic Features**
- ▼ **Unpredictable Time or Money Costs as You Scale**

Consider "open source" or "commercial" options to outweigh these cons.

Commercial Ingress Controllers

Licensed products that are designed to support large production deployments.

Pros

Top reasons a commercial Ingress controller could be right for you

- ▲ **Large Feature Set**
- ▲ **Scalable Time Saver**
- ▲ **Reliable and Supported**

Ideal when...
You need to reduce management complexity and accelerate time to market for new product features.

Cons

Top reasons a commercial Ingress controller could be wrong for you

- ▼ **Slower Feature Velocity**
- ▼ **Requires Monetary Investment**

Consider “open source” or “default” options to outweigh these cons.

Improve Security and Compliance with NGINX Ingress Controller

The NGINX Plus-based edition unlocks five use cases that are critical for keeping your apps and customers safe.

01 **Secure the edge**

02 **Centralize authentication and authorization**

03 **Implement end-to-end encryption**

04 **Get timely and proactive patch notifications**

05 **Be FIPS compliant**

Learn how German automotive giant Audi secured their Red Hat OpenShift apps in **Audi Future-Proofs Tech Vision and App Innovation with NGINX.**

Better Application Performance and Resiliency with NGINX Ingress Controller

The NGINX Plus-based edition unlocks five use cases that help you deliver on the promises of Kubernetes.

01 **Get live monitoring**

02 **Detect and resolve failures faster**

03 **Reconfigure with zero restarts**

04 **Thoroughly test new features and deployments**

05 **Resolve support needs quickly**

Learn how business text messaging company Zipwhip accomplished 99.99% uptime for their SaaS apps in **Strengthen Security and Traffic Visibility on Amazon EKS with NGINX.**

Ready to Learn More?

Read the 4-Part blog series

Part 1: Identify Your Requirements

Identify your Ingress controller requirements, including the problems you want it to solve and whether you'll resource it with time, money, or both!

Part 2: Risks and Future-Proofing

Recognize the risks you might introduce by selecting the wrong Ingress controller, and the key factors that can future-proof your selection.

Part 3: Open Source vs. Default vs. Commercial

Narrow down your Ingress controller selection by delving into the pros and cons for the three categories: open source, default, and commercial.

Part 4: NGINX Ingress Controller Options

Discover which NGINX Ingress controller is best for you, based on authorship, development philosophy, production readiness, security, and support.

NGINX
Part of F5

Contact us to discuss your use cases.