NGINX

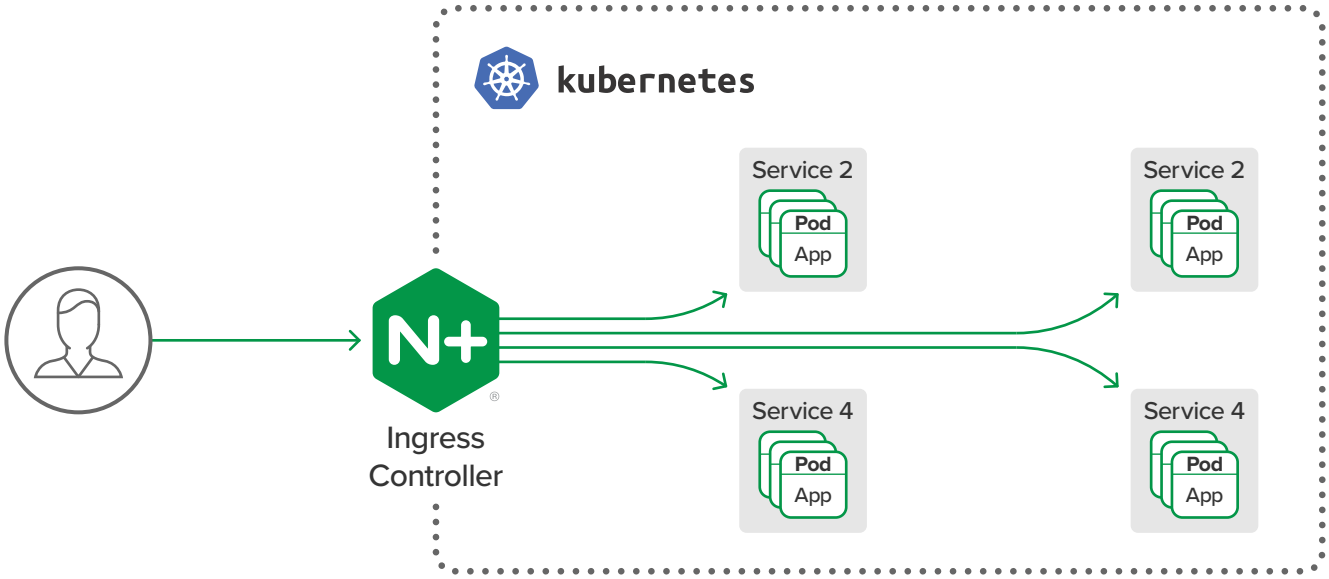
NGINX Ingress Controller for Kubernetes

High-Velocity Application Delivery for Kubernetes

Applications running on Kubernetes need a proven, production-grade application delivery solution. The NGINX Ingress Controller combines the trusted NGINX Open Source and NGINX Plus software load balancers with simplified configuration based on standard Kubernetes Ingress or custom NGINX Ingress resources, to ensure that applications in your Kubernetes cluster are delivered reliably, securely, and at high velocity.

Why Use the NGINX Ingress Controller for Kubernetes?

- **Reduced complexity** – Configure NGINX or NGINX Plus using standard Kubernetes Ingress resources or leverage new NGINX Ingress resources instead of writing native NGINX configuration.
- **Advanced load balancing** – Support blue-green deployments, canary releases, A/B testing, and circuit breakers via advanced load balancing and request routing features available in NGINX Ingress resources.
- **Multiprotocol support** – Deliver HTTP, HTTP/2, gRPC, TCP, and UDP applications.
- **Observability** – Take advantage of tracing support (through OpenTracing), detailed logging capabilities, and native Prometheus integration, as well as a wide range of real-time statistics about application traffic flow (for NGINX Plus).
- **Security** – Optimize SSL/TLS termination performance with configurable encryption (including wildcard certificates) and secure your applications using JWT authentication (for NGINX Plus).
- **Self-service and multi-tenancy** – Leverage RBAC and cross-namespace capabilities of NGINX Ingress resources, which allow for clear demarcation and delegation of application delivery component management across different teams.
- **Production ready**–Feel confident with a stable, reliable Ingress Controller tested by NGINX and covered by 24x7 support for NGINX Plus customers.



Key Capabilities

Advanced Request Routing

The custom NGINX Ingress resources (`VirtualServer` and `VirtualServerRoute`) provide more control over traffic classification and routing decisions than the standard Kubernetes Ingress resource. NGINX Ingress resources support:

- Request routing based on the request URI, headers, cookies, and method
- Splitting traffic among multiple versions of an application based on weights

```
upstreams:
- name: webapp-v1
  service: webapp-svc-v1
  port: 80
- name: webapp-v2
  service: webapp-svc-v2
  port: 80
routes:
- path: /
  splits:
  - weight: 90
    action:
      pass: webapp-v1
  - weight: 10
    action:
      pass: webapp-v2
```

Traffic splitting in NGINX Ingress resources

```
upstreams:
- name: webapp-v1
  service: webapp-svc-v1
  port: 80
- name: webapp-v2
  service: webapp-svc-v2
  port: 80
routes:
- path: /
  matches:
  - conditions:
    - cookie: debug
      value: true
    action:
      pass: webapp-v2
  action:
    pass: webapp-v1
```

Request routing based on a cookie in NGINX Ingress resources

RBAC and Multi-Tenancy

The NGINX Ingress resources allow for the various traffic configuration components to be delegated to different teams while still enabling global configuration to be enforced for all exposed services. This allows, for example, for the security ops team to enforce TLS settings for all exposed services on one resource (`VirtualServer`) while delegating upstream configuration to one or multiple application owners by placing the corresponding resource (`VirtualServerRoute`) in a separate Kubernetes namespace. Such a delegation is enforced using Kubernetes' native RBAC mechanisms.

```
apiVersion: k8s.nginx.org/v1
kind: VirtualServer
metadata:
  name: api-fe
  namespace: frontend-ns
spec:
  host: api.example.com
  tls:
    secret: api-ssl-secret
  routes:
  - path: /games/api
    route: games-ns/games-route
  - path: /stats/api
    route: stats-ns/stats-route
```

`VirtualServer` *defines TLS settings for a host and delegates route definitions to the referenced* `VirtualServerRoute` *resources*

```
apiVersion: k8s.nginx.org/v1
kind: VirtualServerRoute
metadata:
  name: games-route
  namespace: games-ns
spec:
  host: api.example.com
  upstreams:
  - name: games
    service: games-svc
    port: 80
  subroutes:
  - path: /games/api
    action:
      pass: games
```

`VirtualServerRoute` *defines routes, but it cannot override the global host TLS settings*

For more information, visit:
nginx.com/products/nginx/kubernetes-ingress-controller
or send us an email at nginx-inquiries@nginx.com

NGINX
Part of F5

Published on 05 February 2020