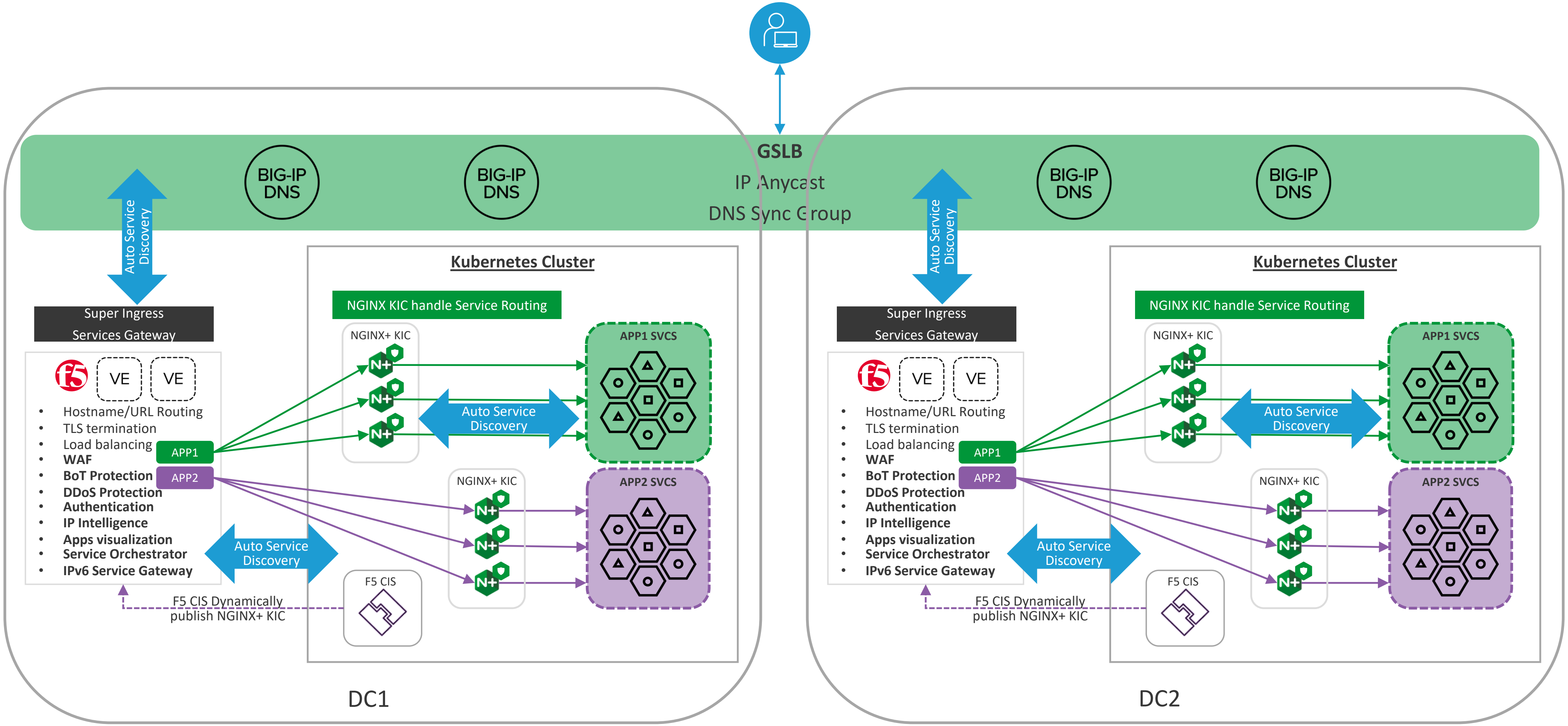
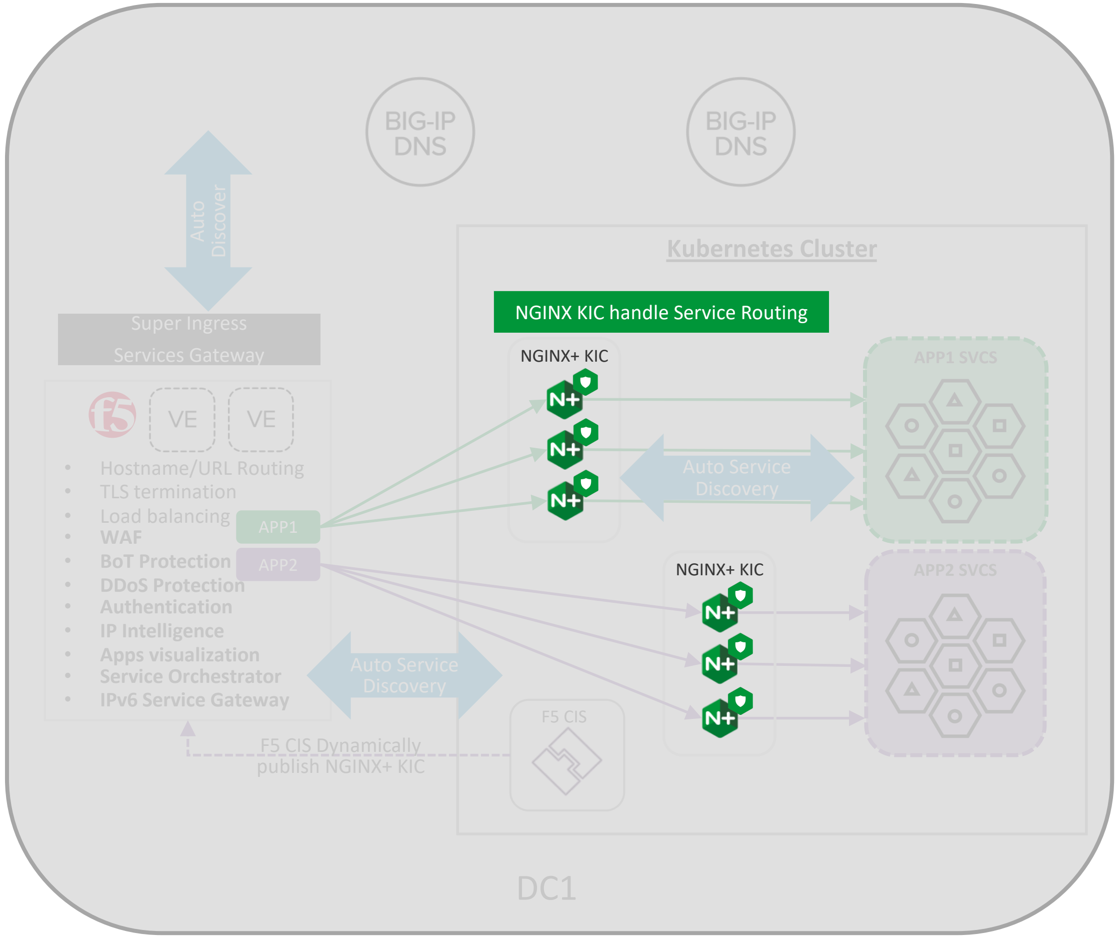
NGINX+ K8s Ingress Controller

Agenda : End-goal with Active-Active K8s Clusters Architecture



Agenda : Day 1



- Hostname/URL Routing
- TLS termination
- Load balancing
- WAF
- BoT Protection
- DDoS Protection
- Authentication
- IP Intelligence
- Apps visualization
- Service Orchestrator
- IPv6 Service Gateway

Chapter 1 : Modern application architecture – F5's vision & solutions

- Reacquaint with F5 Solutions
- Modern Infrastructure with F5 BIG-IP
- Modern Application with NGINX
- Edge Delivery Platform with F5 Distributed Cloud

Chapter 2 : K8s fundamentals & networking (concept + Lab)

- Containers and Pods
- YAML Basics – How to read k8s manifests
- K8s Deployment
- K8s Networking: ClusterIP, Services, LB, NodePort

Chapter 3: K8s and NGINX Ingress Controller (concept + Lab)

- What is Ingress Controller?
- How does NGINX+ look like as Ingress Controller?
- NGINX+ value, positioning & strength

What is the Ingress?

The Ingress is a Kubernetes resource that lets you configure an HTTP load balancer for applications running on Kubernetes, represented by one or more Services. Such a load balancer is necessary to deliver those applications to clients outside of the Kubernetes cluster.

The Ingress resource supports the following features:

- **Content-based routing**:
  - *Host-based routing*. For example, routing requests with the host header foo.example.com to one group of services and the host header bar.example.com to another group.
  - *Path-based routing*. For example, routing requests with the URI that starts with /serviceA to service A and requests with the URI that starts with /serviceB to service B.
- **TLS/SSL termination** for each hostname, such as `foo.example.com`.

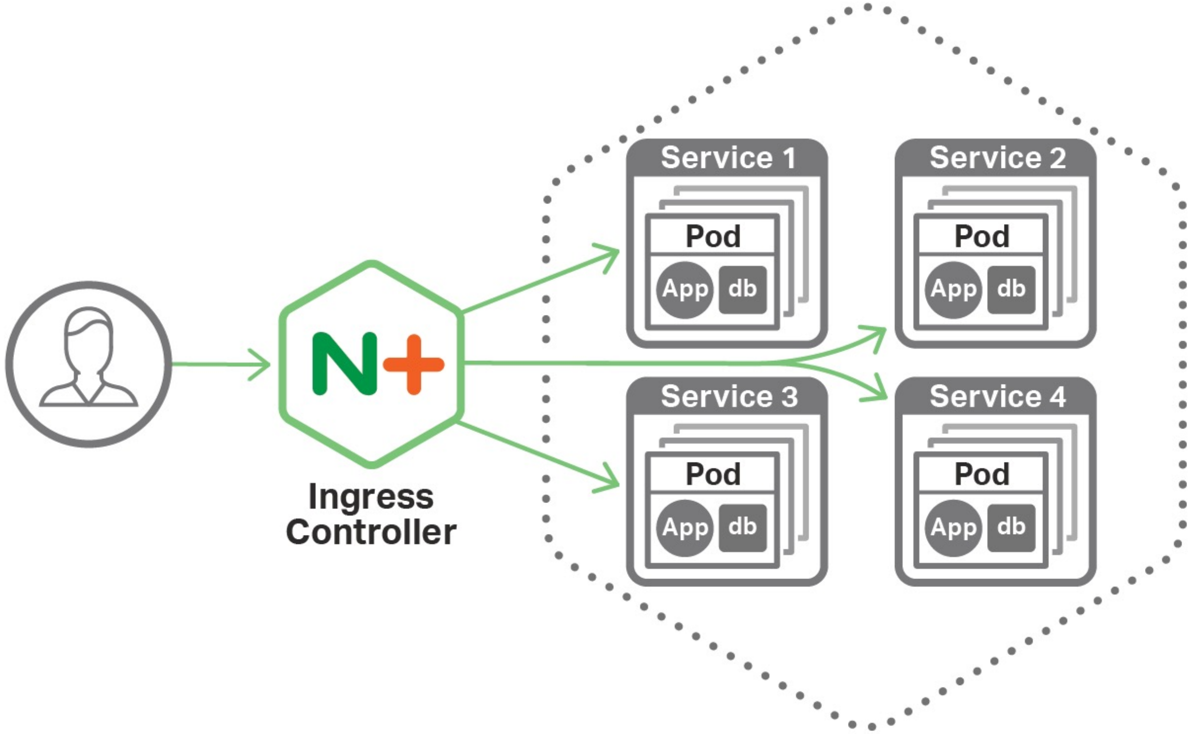
https://docs.nginx.com/nginx-ingress-controller/intro/overview/

What is the Ingress Controller?

The Ingress Controller is an application that runs in a cluster and configures an HTTP load balancer according to Ingress resources. The load balancer can be a software load balancer running in the cluster or a hardware or cloud load balancer running externally. Different load balancers require different Ingress Controller implementations.

In the case of NGINX, the Ingress Controller is deployed in a pod along with the load balancer.

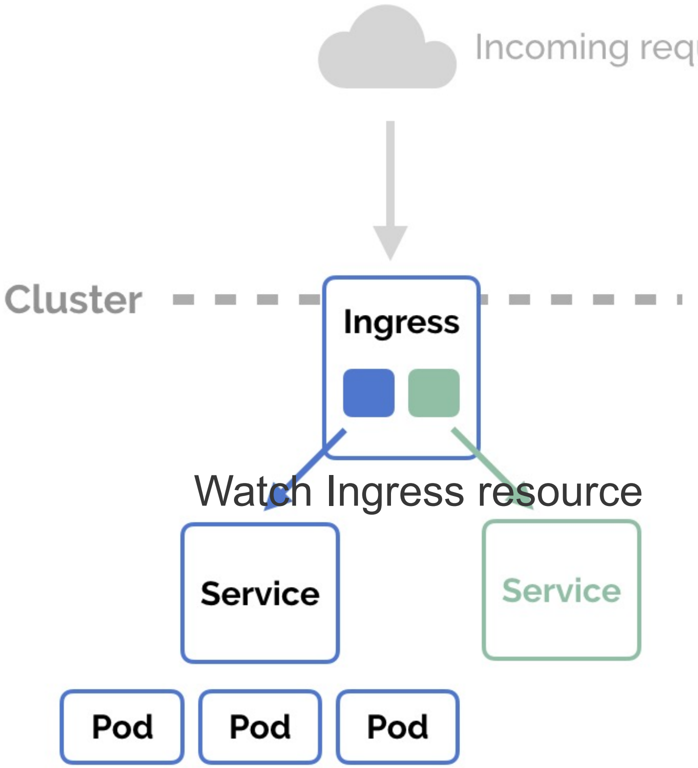
NGINX+ Ingress Controller (KIC)

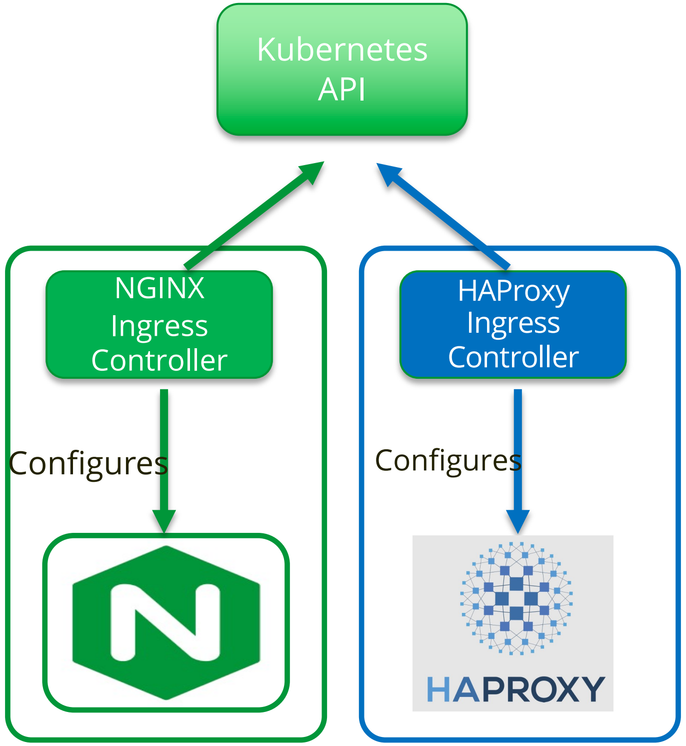


Key Benefits not found in NGINX OSS:

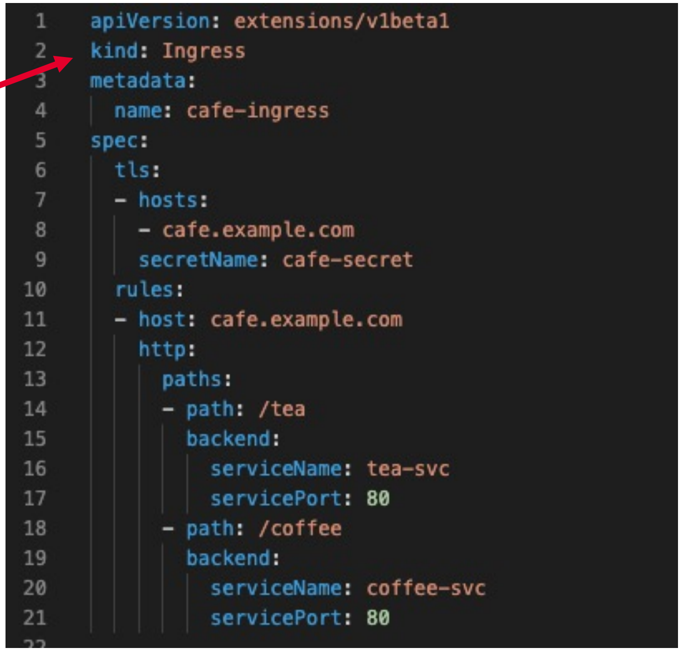
- Flexible and granular controls for complex configurations (CRD)
- Advanced Load balancing
- Dynamic reconfiguration
- JWT authentication
- Session persistence
- Real-time monitoring
- 24x7 support
- Optional NGINX App Protect WAF & DoS

Ingress / Ingress Controller

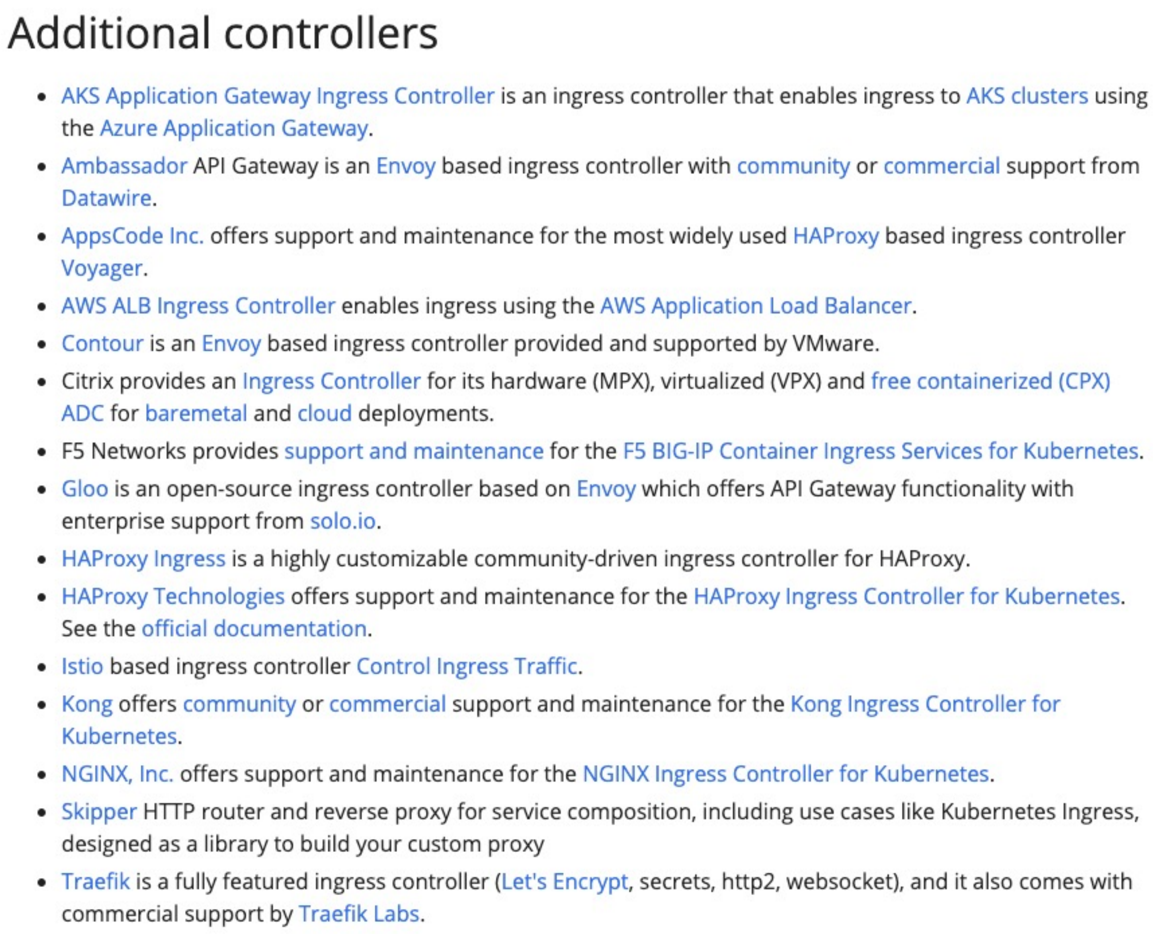




Ingress resource

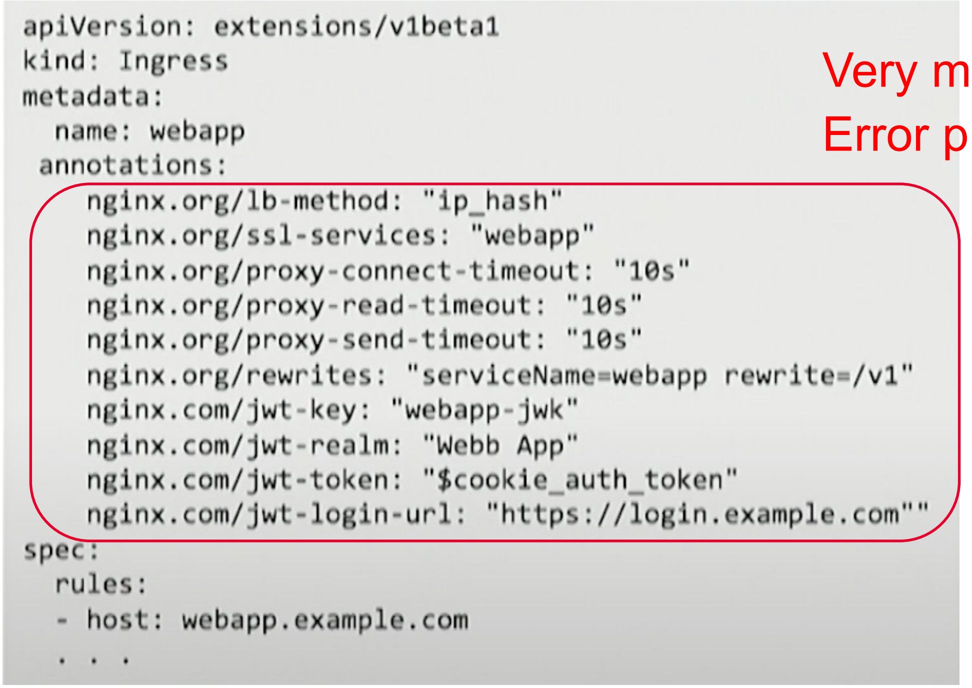


```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: cafe-ingress
spec:
  tls:
  - hosts:
    - cafe.example.com
    secretName: cafe-secret
  rules:
  - host: cafe.example.com
    http:
      paths:
      - path: /tea
        backend:
          serviceName: tea-svc
          servicePort: 80
      - path: /coffee
        backend:
          serviceName: coffee-svc
          servicePort: 80
```



Additional controllers

- AKS Application Gateway Ingress Controller is an ingress controller that enables ingress to AKS clusters using the Azure Application Gateway.
- Ambassador API Gateway is an Envoy based ingress controller with community or commercial support from Datawire.
- AppsCode Inc. offers support and maintenance for the most widely used HAProxy based ingress controller Voyager.
- AWS ALB Ingress Controller enables ingress using the AWS Application Load Balancer.
- Contour is an Envoy based ingress controller provided and supported by VMware.
- Citrix provides an Ingress Controller for its hardware (MPX), virtualized (VPX) and free containerized (CPX) ADC for baremetal and cloud deployments.
- F5 Networks provides support and maintenance for the F5 BIG-IP Container Ingress Services for Kubernetes.
- Gloo is an open-source ingress controller based on Envoy which offers API Gateway functionality with enterprise support from solo.io.
- HAProxy Ingress is a highly customizable community-driven ingress controller for HAProxy.
- HAProxy Technologies offers support and maintenance for the HAProxy Ingress Controller for Kubernetes. See the official documentation.
- Istio based ingress controller Control Ingress Traffic.
- Kong offers community or commercial support and maintenance for the Kong Ingress Controller for Kubernetes.
- NGINX, Inc. offers support and maintenance for the NGINX Ingress Controller for Kubernetes.
- Skipper HTTP router and reverse proxy for service composition, including use cases like Kubernetes Ingress, designed as a library to build your custom proxy
- Traefik is a fully featured ingress controller (Let's Encrypt, secrets, http2, websocket), and it also comes with commercial support by Traefik Labs.



```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: webapp
 annotations:
    nginx.org/lb-method: "ip_hash"
    nginx.org/ssl-services: "webapp"
    nginx.org/proxy-connect-timeout: "10s"
    nginx.org/proxy-read-timeout: "10s"
    nginx.org/proxy-send-timeout: "10s"
    nginx.org/rewrites: "serviceName=webapp rewrite=/v1"
    nginx.com/jwt-key: "webapp-jwk"
    nginx.com/jwt-realm: "Webb App"
    nginx.com/jwt-token: "$cookie_auth_token"
    nginx.com/jwt-login-url: "https://login.example.com""
spec:
  rules:
  - host: webapp.example.com
  . . .
```

Very messy
Error prone

Custom Resource Definition – CRD

NGINX – VirtualServer/VirtualServerRoute
Contour – HTTPProxy
Traefik – IngressRoute

Two Challenges at Scale

Complex Applications

How can you provide the advanced capabilities that complex applications require?

Multitenancy (Teams)

How can multiple teams and applications share a Container environment safely and securely?

Meet your personas

David

Super-NetOps

Olivia

DevOps

Iron

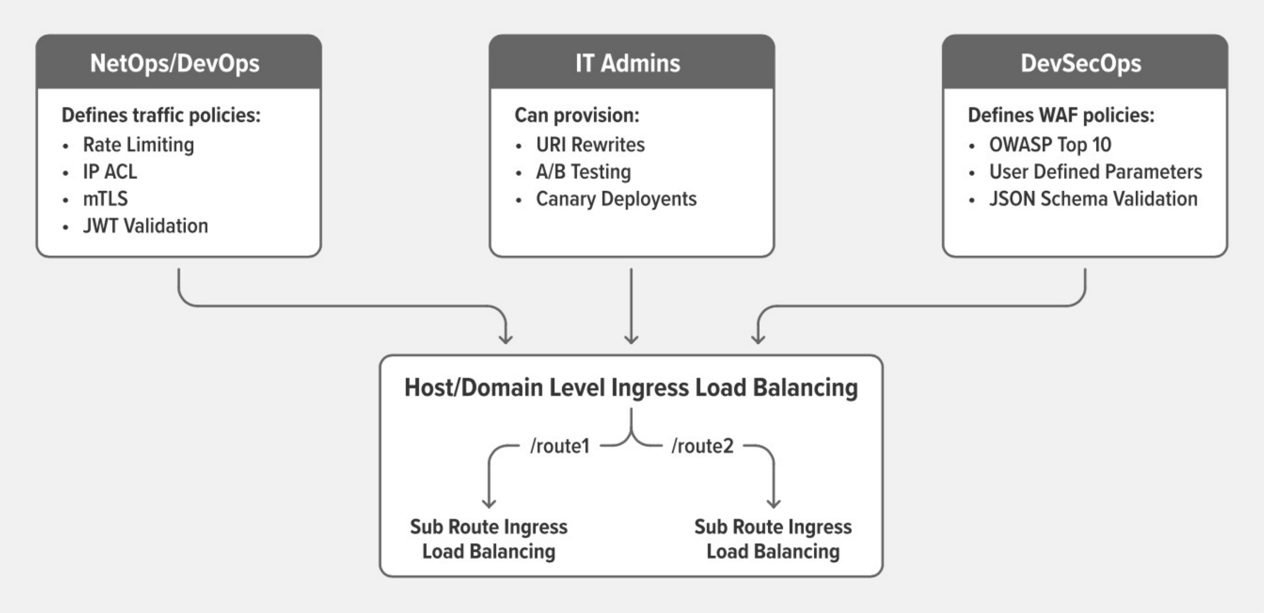
Platform Team

Chris

SecOps

Dwayne

Cloud Architect



NetOps/DevOps

Defines traffic policies:

- Rate Limiting
- IP ACL
- mTLS
- JWT Validation

IT Admins

Can provision:

- URI Rewrites
- A/B Testing
- Canary Deployents

DevSecOps

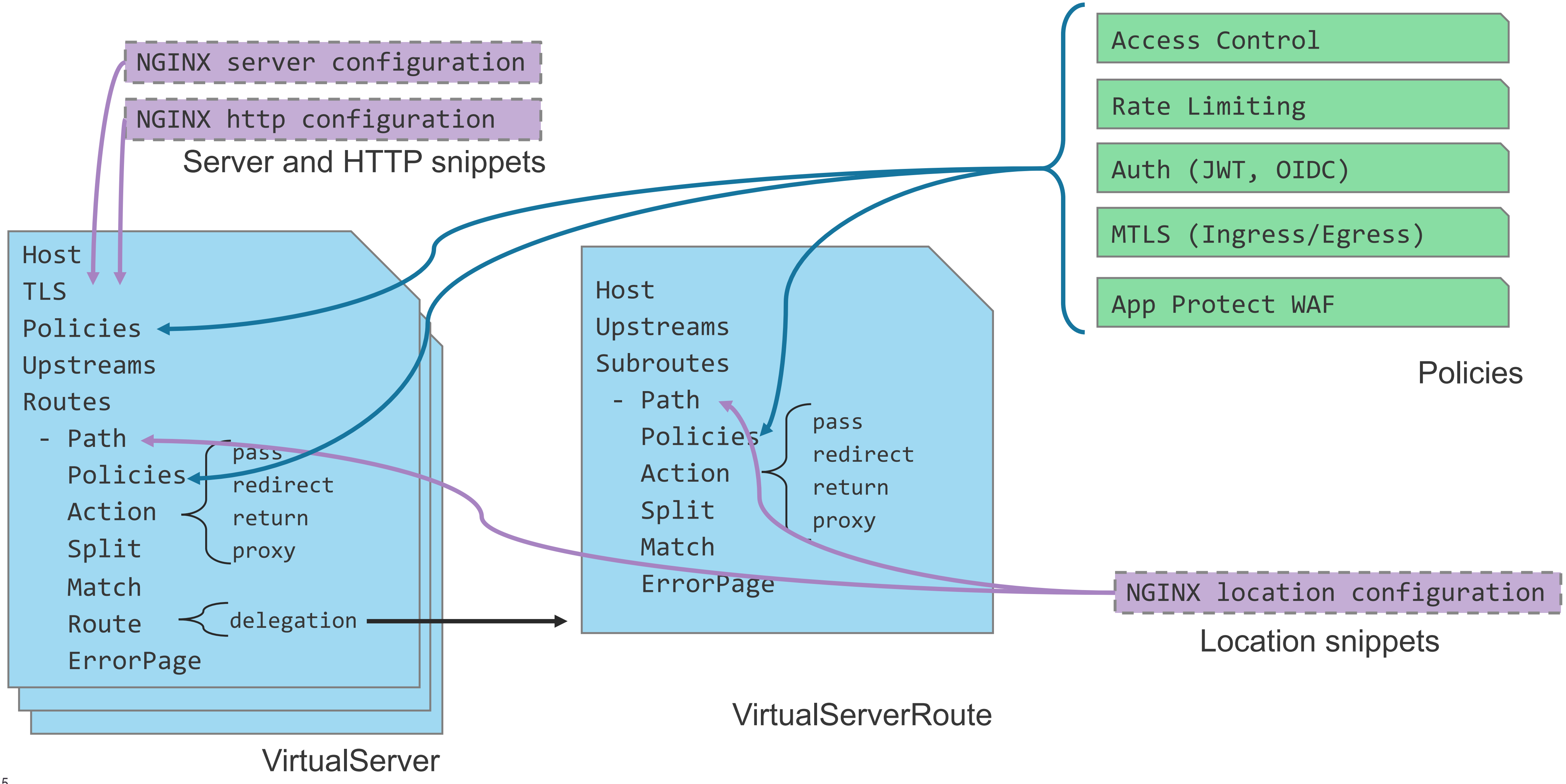
Defines WAF policies:

- OWASP Top 10
- User Defined Parameters
- JSON Schema Validation

Host/Domain Level Ingress Load Balancing

/route1
/route2
Sub Route Ingress Load Balancing
Sub Route Ingress Load Balancing

NGINX+ Ingress Resources – Rich Capabilities



NGINX server configuration

NGINX http configuration

Server and HTTP snippets

Access Control

Rate Limiting

Auth (JWT, OIDC)

MTLS (Ingress/Egress)

App Protect WAF

Policies

```
Host
TLS
Policies
Upstreams
Routes
 - Path
   Policies
   Action    pass
             redirect
             return
             proxy
   Split
   Match
   Route     delegation
   ErrorPage
```

VirtualServer

```
Host
Upstreams
Subroutes
 - Path
   Policies
   Action    pass
             redirect
             return
             proxy
   Split
   Match
   ErrorPage
```

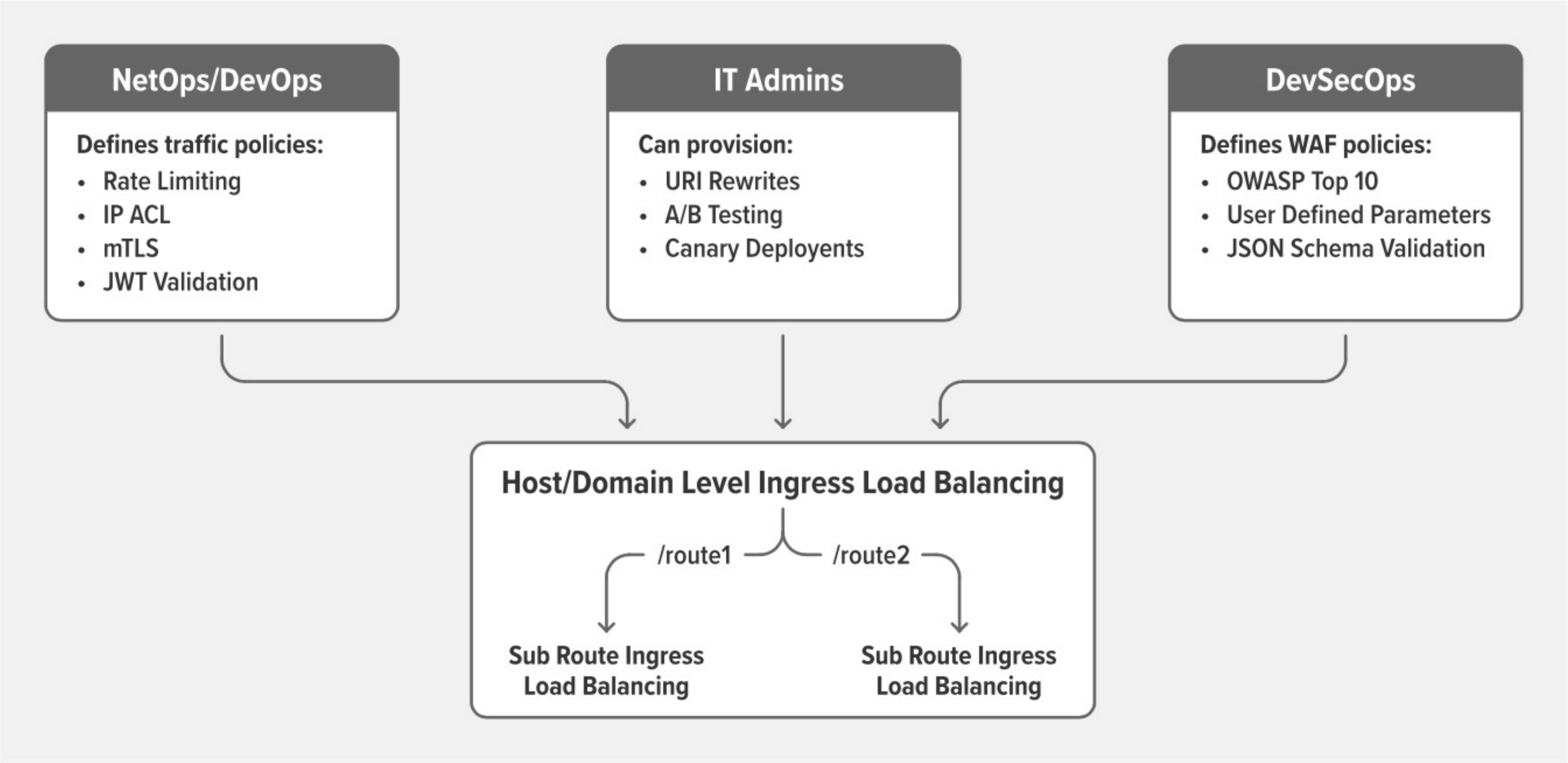
VirtualServerRoute

NGINX location configuration

Location snippets

NGINX+ Ingress Controller

BUILT FOR ALL PERSONAS



NetOps/DevOps

Defines traffic policies:

- Rate Limiting
- IP ACL
- mTLS
- JWT Validation

IT Admins

Can provision:

- URI Rewrites
- A/B Testing
- Canary Deployents

DevSecOps

Defines WAF policies:

- OWASP Top 10
- User Defined Parameters
- JSON Schema Validation

Host/Domain Level Ingress Load Balancing

/route1

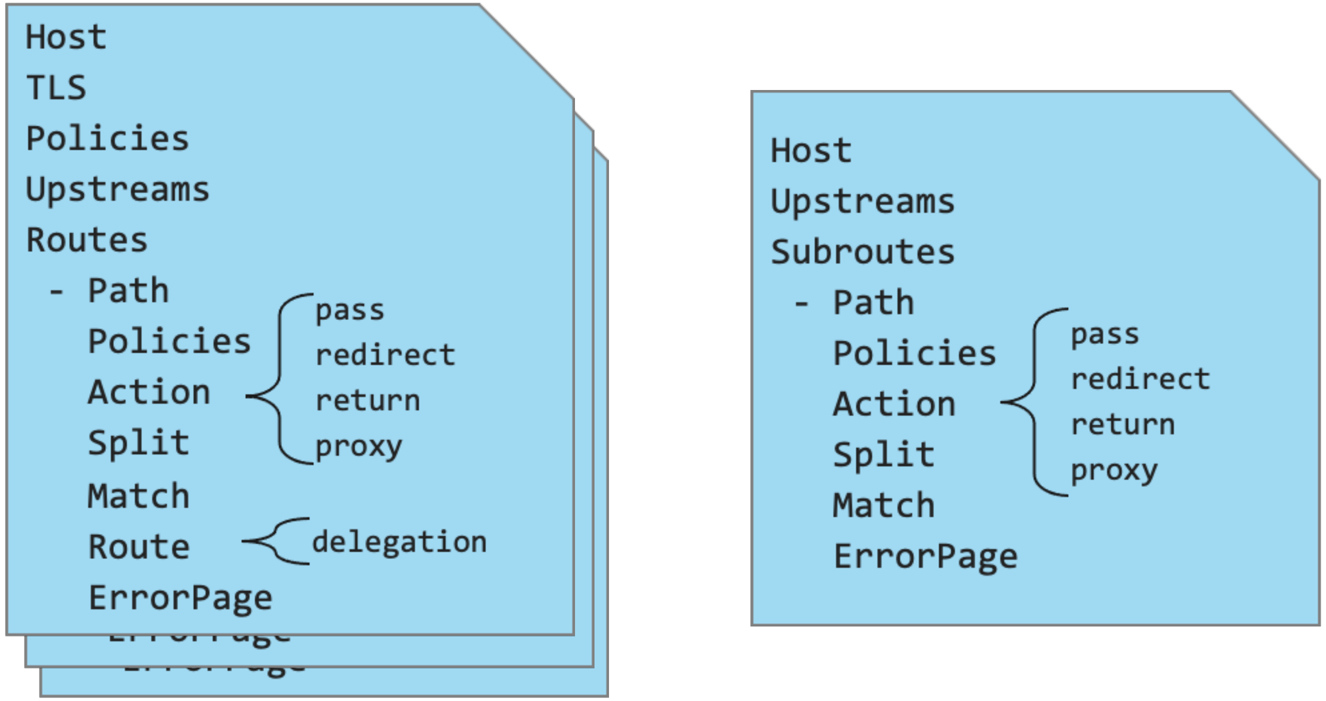
/route2

Sub Route Ingress Load Balancing

Sub Route Ingress Load Balancing

NGINX+ Ingress Resources – Distributed Configuration

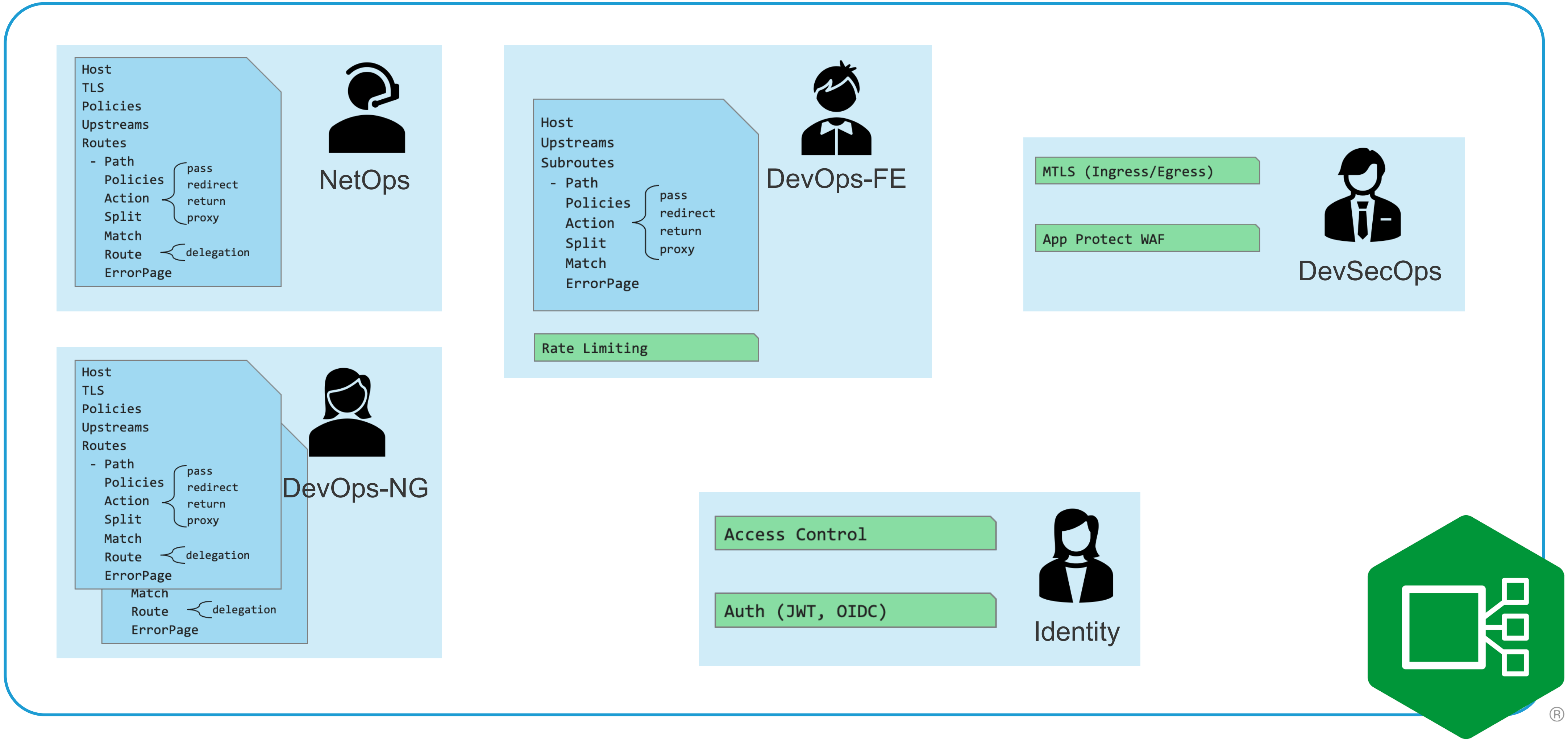
- Access Control
- Rate Limiting
- Auth (JWT, OIDC)
- MTLS (Ingress/Egress)
- App Protect WAF



Host
TLS
Policies
Upstreams
Routes
  - Path
    Policies
    Action { pass, redirect, return, proxy }
    Split
    Match
    Route { delegation }
    ErrorPage

Host
Upstreams
Subroutes
  - Path
    Policies
    Action { pass, redirect, return, proxy }
    Split
    Match
    ErrorPage

NGINX+ Ingress Resources – Distributed Configuration



```
Host
TLS
Policies
Upstreams
Routes
 - Path
   Policies     pass
   Action       redirect
   Split        return
                proxy
   Match
   Route      delegation
   ErrorPage
```

NetOps

```
Host
Upstreams
Subroutes
 - Path
   Policies     pass
   Action       redirect
   Split        return
   Match        proxy
   ErrorPage
```

Rate Limiting

DevOps-FE

MTLS (Ingress/Egress)

App Protect WAF

DevSecOps

```
Host
TLS
Policies
Upstreams
Routes
 - Path
   Policies     pass
   Action       redirect
   Split        return
                proxy
   Match
   Route      delegation
   ErrorPage
   Match
   Route      delegation
   ErrorPage
```

DevOps-NG

Access Control

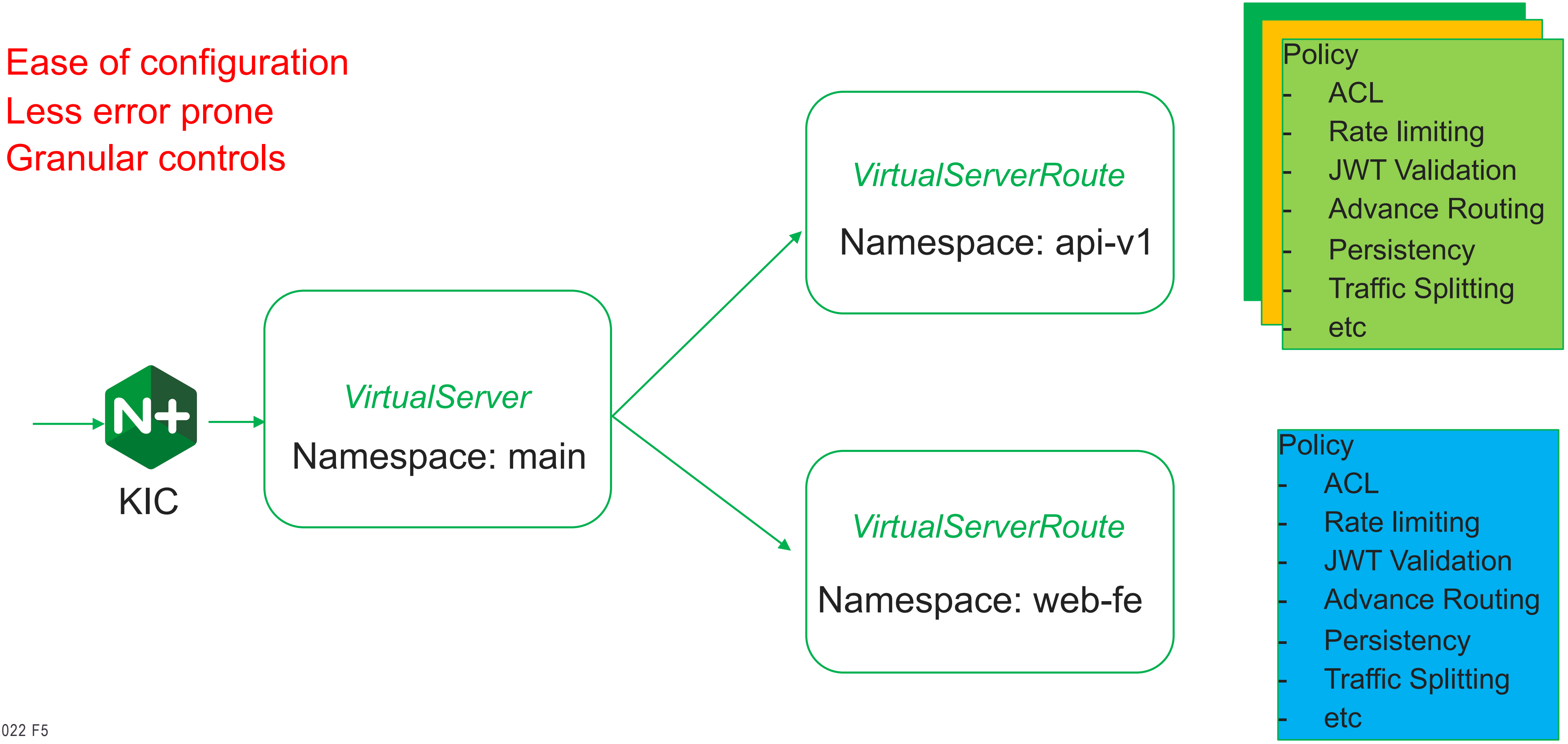
Auth (JWT, OIDC)

Identity

VirtualServer & VirtualServerRoute

VirtualServer and VirtualServerRoute are new load balancing resources, enable use cases not supported with a typical Ingress resource (traffic splitting and advanced content-based routing), and allow different teams to apply different policies

- Ease of configuration
- Less error prone
- Granular controls



KIC

VirtualServer

Namespace: main

VirtualServerRoute

Namespace: api-v1

VirtualServerRoute

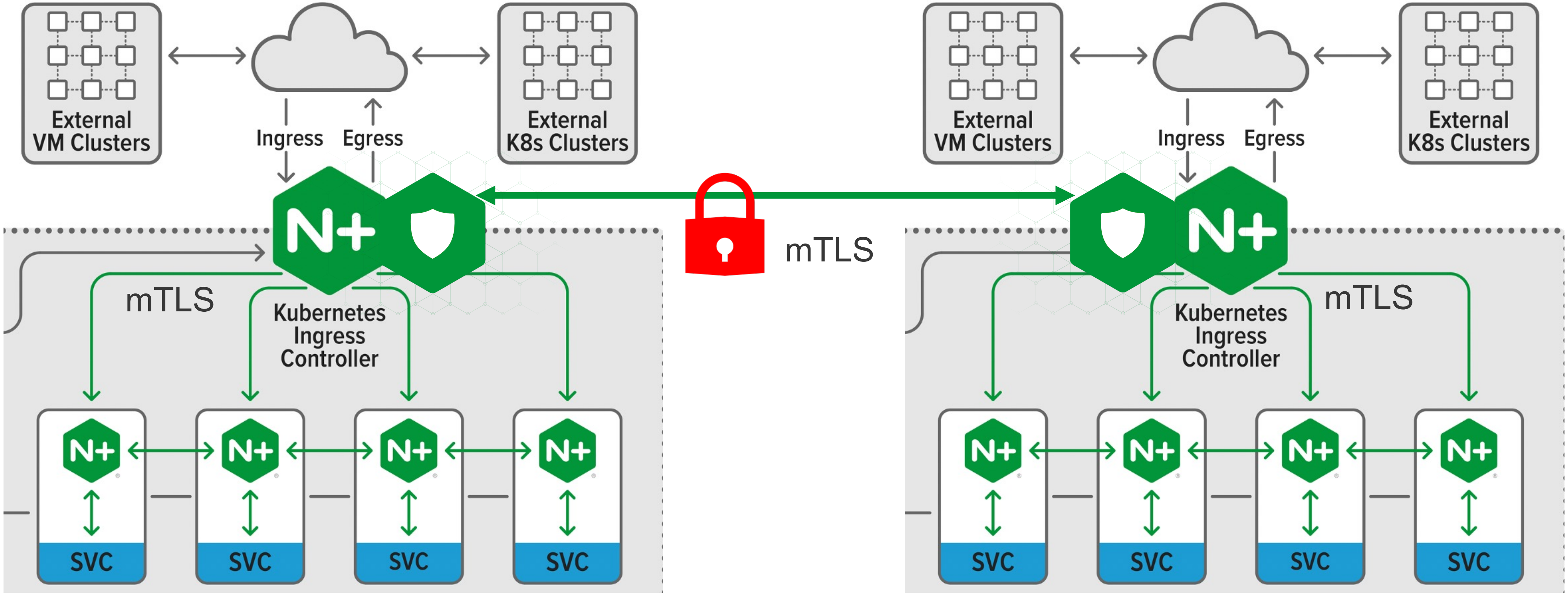
Namespace: web-fe

Policy
- ACL
- Rate limiting
- JWT Validation
- Advance Routing
- Persistency
- Traffic Splitting
- etc

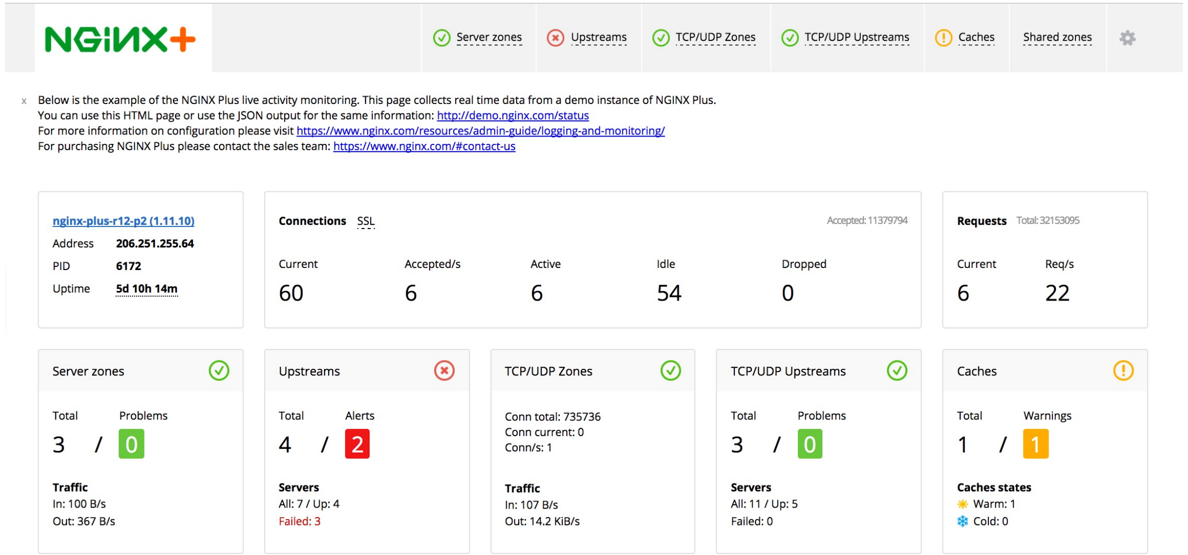
Policy
- ACL
- Rate limiting
- JWT Validation
- Advance Routing
- Persistency
- Traffic Splitting
- etc

MTLS Use Case: Multi-Cluster Edge Security

NSM and N+ KIC and NAP: End-to-end encryption between environments with edge firewall



NGINX+ – Real Time Monitoring



Below is the example of the NGINX Plus live activity monitoring. This page collects real time data from a demo instance of NGINX Plus.
You can use this HTML page or use the JSON output for the same information: http://demo.nginx.com/status
For more information on configuration please visit https://www.nginx.com/resources/admin-guide/logging-and-monitoring/
For purchasing NGINX Plus please contact the sales team: https://www.nginx.com/#contact-us

nginx-plus-r12-p2 (1.11.10)
Address 206.251.255.64
PID 6172
Uptime 5d 10h 14m

Connections SSL — Accepted: 11379794

Current	Accepted/s	Active	Idle	Dropped
60	6	6	54	0

Requests Total: 32153095

Current	Req/s
6	22

Server zones ✓
Total 3 / Problems 0
Traffic
In: 100 B/s
Out: 367 B/s

Upstreams ☒
Total 4 / Alerts 2
Servers
All: 7 / Up: 4
Failed: 3

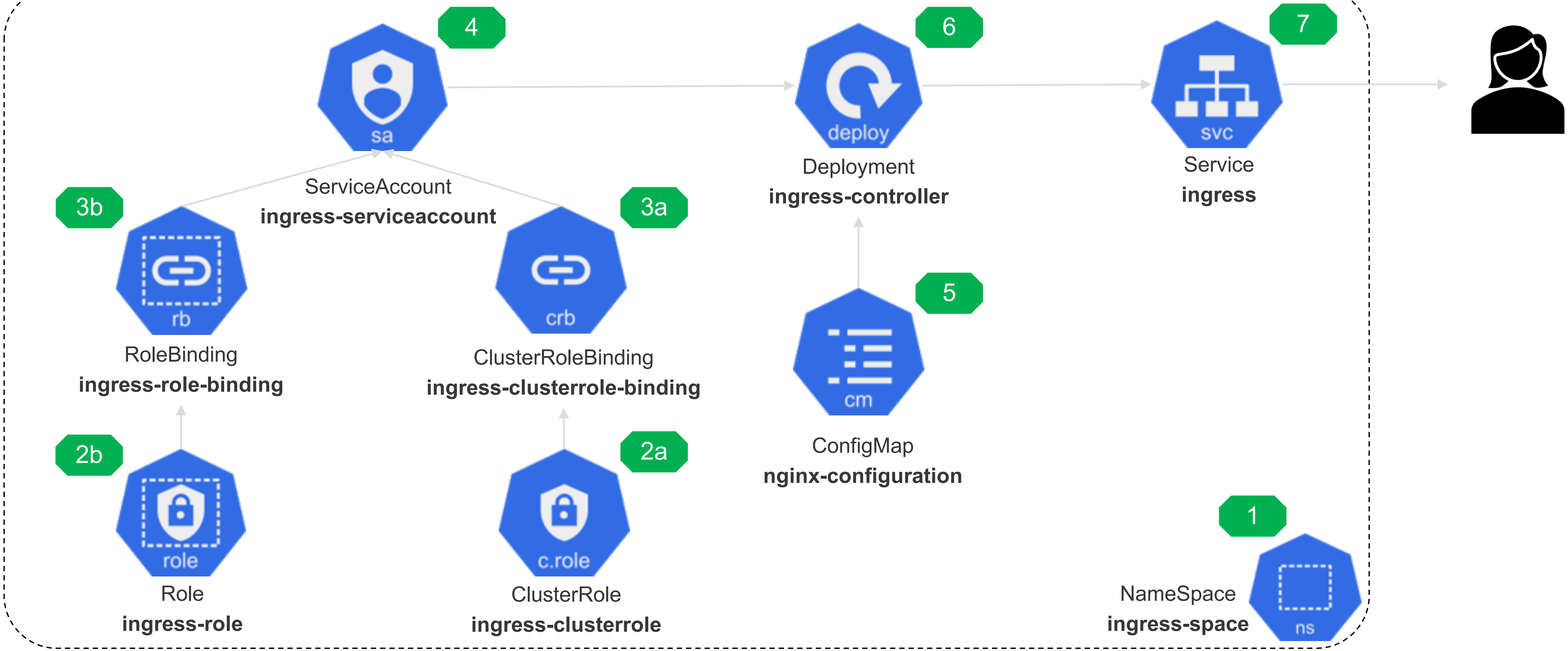
TCP/UDP Zones ✓
Conn total: 735736
Conn current: 0
Conn/s: 1
Traffic
In: 107 B/s
Out: 14.2 KiB/s

TCP/UDP Upstreams ✓
Total 3 / Problems 0
Servers
All: 11 / Up: 5
Failed: 0

Caches !
Total 1 / Warnings 1
Caches states
Warm: 1
Cold: 0

- NGINX+ exposes key metrics about traffic flow through the cluster
- Uses the extended status module and NGINX+ dashboard.
- Real-time metrics and statistics for better insights and visibility into applications, load balancer performance and traffic flow.

(Typical) Ingress Controller Deployment Flow



Lab 3.1 – Deploy App in AWS EKS (K8s)

Lab 3.1 – Deploy Application in AWS EKS (K8s)

Lab Tasks

3.1.1) Deploy the application (Arcadia) in K8s

3.1.2) Verify the deployed pods

3.1.3) Verify the deployed services

Lab 3.2 – Deploy NGINX+ KIC

Lab 3.2 – NGINX+ K8s Ingress Controller

<u>Lab Tasks</u>

3.2.1) NGINX+ K8s Ingress Installation

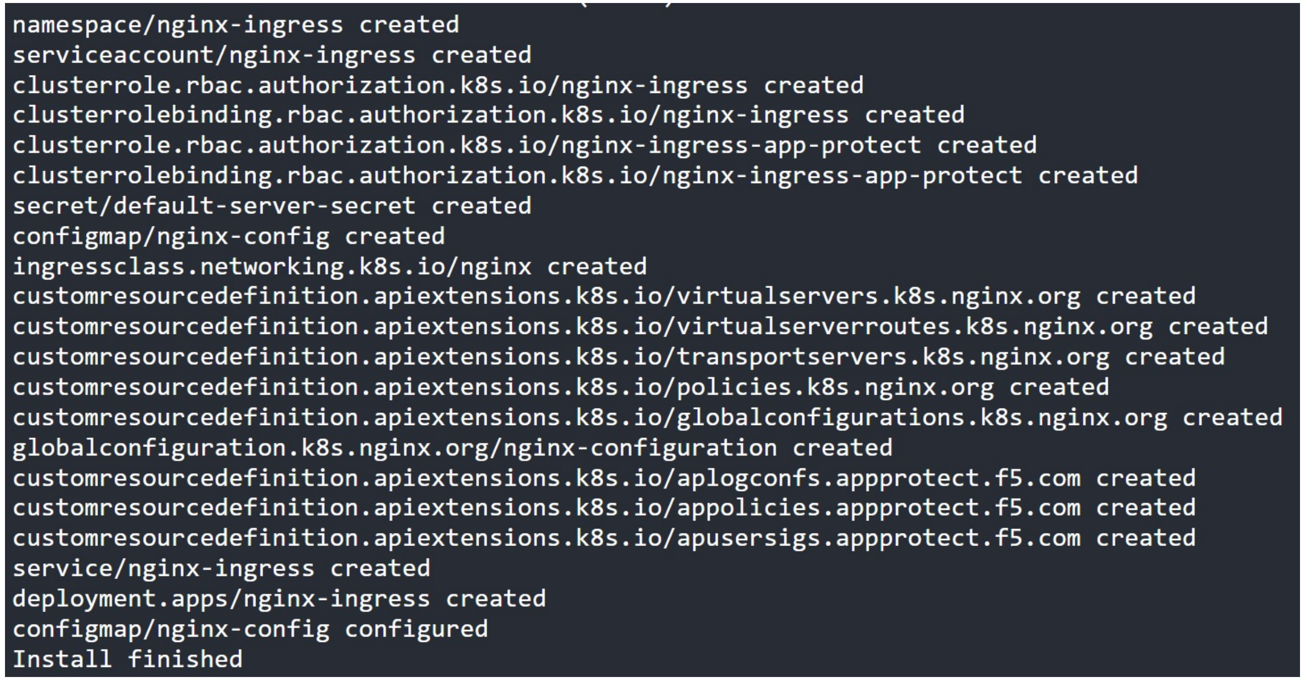
3.2.2) Review each installation manifest file

3.2.3) Expose the NGINX+ Ingress Dashboard Service

3.2.4) Publish the application (Arcadia) to the Internet

3.2.5) Enable HTTPS and Health Checks Monitoring

3.2.6) Using VirtualServer and VirtualServerRoute CRDs



```
namespace/nginx-ingress created
serviceaccount/nginx-ingress created
clusterrole.rbac.authorization.k8s.io/nginx-ingress created
clusterrolebinding.rbac.authorization.k8s.io/nginx-ingress created
clusterrole.rbac.authorization.k8s.io/nginx-ingress-app-protect created
clusterrolebinding.rbac.authorization.k8s.io/nginx-ingress-app-protect created
secret/default-server-secret created
configmap/nginx-config created
ingressclass.networking.k8s.io/nginx created
customresourcedefinition.apiextensions.k8s.io/virtualservers.k8s.nginx.org created
customresourcedefinition.apiextensions.k8s.io/virtualserverroutes.k8s.nginx.org created
customresourcedefinition.apiextensions.k8s.io/transportservers.k8s.nginx.org created
customresourcedefinition.apiextensions.k8s.io/policies.k8s.nginx.org created
customresourcedefinition.apiextensions.k8s.io/globalconfigurations.k8s.nginx.org created
globalconfiguration.k8s.nginx.org/nginx-configuration created
customresourcedefinition.apiextensions.k8s.io/aplogconfs.appprotect.f5.com created
customresourcedefinition.apiextensions.k8s.io/appolicies.appprotect.f5.com created
customresourcedefinition.apiextensions.k8s.io/apusersigs.appprotect.f5.com created
service/nginx-ingress created
deployment.apps/nginx-ingress created
configmap/nginx-config configured
Install finished
```

Additional Info for NGINX+ KIC Installation

https://docs.nginx.com/nginx-ingress-controller/installation/installation-with-manifests/

https://github.com/nginxinc/kubernetes-ingress

Prerequisites:

Make sure you have access to the NGINX Plus Ingress Controller image:

1. For NGINX Plus Ingress Controller, see here for details on how to pull the image from the F5 Docker registry: https://docs.nginx.com/nginx-ingress-controller/installation/pulling-ingress-controller-image

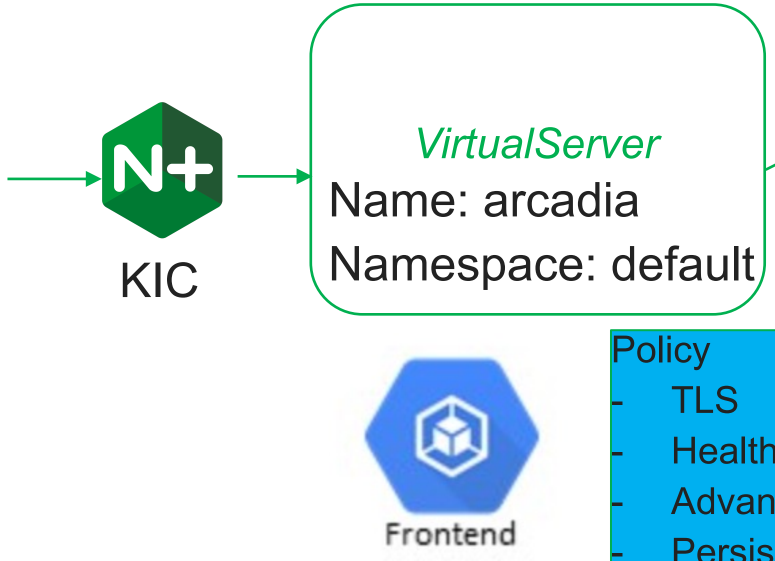
2. To pull from the F5 Container registry in your Kubernetes cluster, configure a docker registry secret using your JWT token from the MyF5 portal by following the instructions from here: https://docs.nginx.com/nginx-ingress-controller/installation/using-the-jwt-token-docker-secret

3. It is also possible to build your own image and push it to your private Docker registry by following the instructions from here: https://docs.nginx.com/nginx-ingress-controller/installation/building-ingress-controller-image

Using VirtualServer & VirtualServerRoute

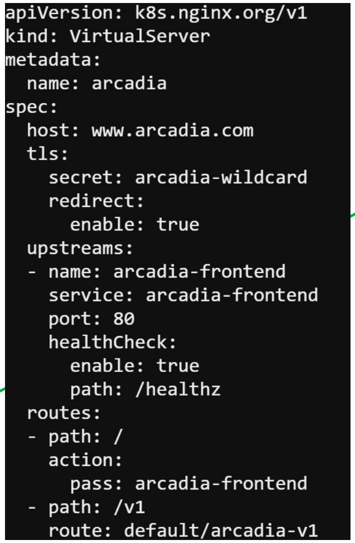
For different team to apply different policies – providing granular controls

- Ease of configuration
- Less error prone
- Granular controls



https://www.arcadia.com/

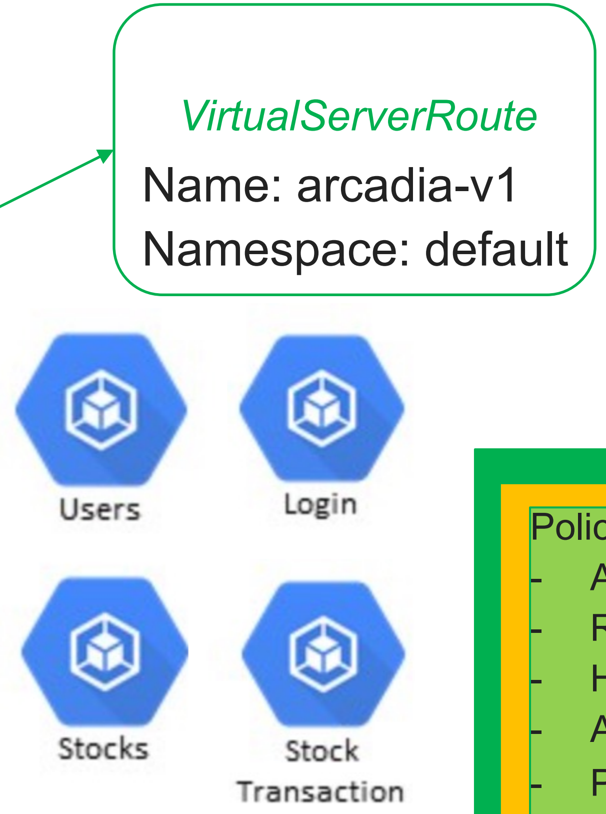
VirtualServer
Name: arcadia
Namespace: default



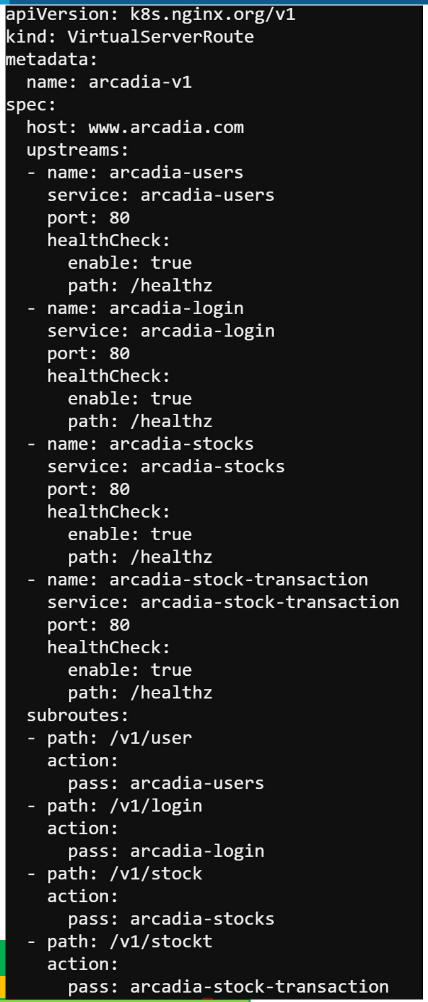
```
apiVersion: k8s.nginx.org/v1
kind: VirtualServer
metadata:
  name: arcadia
spec:
  host: www.arcadia.com
  tls:
    secret: arcadia-wildcard
    redirect:
      enable: true
  upstreams:
  - name: arcadia-frontend
    service: arcadia-frontend
    port: 80
    healthCheck:
      enable: true
      path: /healthz
  routes:
  - path: /
    action:
      pass: arcadia-frontend
  - path: /v1
    route: default/arcadia-v1
```

Policy
- TLS
- HealthCheck
- Advance Routing
- Persistency
- Traffic Splitting

https://www.arcadia.com/v1/user
https://www.arcadia.com/v1/login
https://www.arcadia.com/v1/stock
https://www.arcadia.com/v1/stockt



VirtualServerRoute
Name: arcadia-v1
Namespace: default



```
apiVersion: k8s.nginx.org/v1
kind: VirtualServerRoute
metadata:
  name: arcadia-v1
spec:
  host: www.arcadia.com
  upstreams:
  - name: arcadia-users
    service: arcadia-users
    port: 80
    healthCheck:
      enable: true
      path: /healthz
  - name: arcadia-login
    service: arcadia-login
    port: 80
    healthCheck:
      enable: true
      path: /healthz
  - name: arcadia-stocks
    service: arcadia-stocks
    port: 80
    healthCheck:
      enable: true
      path: /healthz
  - name: arcadia-stock-transaction
    service: arcadia-stock-transaction
    port: 80
    healthCheck:
      enable: true
      path: /healthz
  subroutes:
  - path: /v1/user
    action:
      pass: arcadia-users
  - path: /v1/login
    action:
      pass: arcadia-login
  - path: /v1/stock
    action:
      pass: arcadia-stocks
  - path: /v1/stockt
    action:
      pass: arcadia-stock-transaction
```

Policy
- ACL
- Rate limiting
- Health Check
- Advance Routing
- Persistency
- Traffic Splitting
- App Protect WAF

Summary

What We Have Learned in This Chapter

- Understand NGINX+ KIC key benefits
- Understand Ingress Controller deployment flow
- Deploy NGINX+ KIC to expose the application in Kubernetes.

K8s Concept

SUMMARY NODEPORT VS INGRESS CONTROLLER

Summary	**NodePort**	**Ingress Controller**
High Level Objectives	To publish the port to be accessed externally	To publish the port to be accessed externally
Components	Kube-proxy & CNI	Kube-proxy & CNI
Advantages (Plus Points)	• Simple way to publish the pods EXTERNALLY outside of the cluster • The pods will be accessible within all nodes in the cluster • Built-in Load-balancing capability to the pods	• Simple way to publish the pods EXTERNALLY outside of the cluster • The pods will be accessible within all nodes in the cluster • **Built-in L7 routing** and Load-balancing capabilities to the pods • All services can use the same KIC
Drawbacks (Negative Points)	• No L7 routing capabilities, traffic will be passed directly to the backend • Require another Load-Balancer to LB between nodes. Otherwise, imbalance traffic might happen • It can only use ports 30000–32767 • If worker nodes IP address change, require changes in the surrounding environment	• *If the ingress controller is fronted by NodePort service, it can only use ports 30000–32767* • *If the ingress controller is fronted by LoadBalancer service, it can use any customed port* • If worker nodes IP address change, require changes in the surrounding environment

Quiz Time